2005

An improved software process management tool: ReMoTe (recursively estimating multi-threaded observation tool enterprise)

Shujiang Xia

AN IMPROVED SOFTWARE PROCESS MANAGEMENT TOOL: REMOTE (RECURSIVELY ESTIMATING MULTI-THREADED OBSERVATION TOOL ENTERPRISE)

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Shujiang Xia

December 2005

AN IMPROVED SOFTWARE PROCESS MANAGEMENT TOOL: REMOTE (RECURSIVELY ESTIMATING MULTI-THREADED OBSERVATION TOOL ENTERPRISE)

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Shujiang Xia

December 2005

Approved by:

[redacted]

Arturo I. Concepcion, Chair, Computer Science — 26 Oct 2005 Date

[redacted]

David Turner

[redacted]

George Georgiou

ABSTRACT

Development of the ReMoTe Software Process Management Tool was done using the Recursive Multi Threaded Software Life-cycle philosophy developed by Simon Scott and Chung-Ping Lin. Using such a philosophy results in an easy to use Web-based project management application tool. ReMoTe allows a software company or a team to organize overall software development into smaller, more manageable components. Once the components of the project are defined, they are then tracked and monitored to insure the project is on schedule. The performance of each team or team members can also be measured against their results. ReMoTe tool allows each member to see the progress of product development as well as a means to resolve development issues between teams. Also, ReMoTe facilitates team communication by offering all team members instant access to information about every component of their project. Additionally, there is no geographic or facility limitation. ReMoTe is able to do this by building interfaces that hides the ReMoTe physical databases such as that mySQL, Oracle or Access, that implements each sub-project. This means that ReMoTe can display the overall project progress in a single ReMoTe thread structure. These sub-projects can also be located anywhere in the world because ReMoTe is Web-based.

ACKNOWLEDGMENTS

I would like to acknowledge the supervision and support of my adviser, the Chair of CSUSB Computer Science Department, Dr. Arturo I Concepcion. He provided me the most valuable suggestions about direction of this project and lots of information and suggestions that extended my understanding in ReMoTe Tool improving the quality of the project.

Subsequently, I would also like to acknowledge my two-committee members, Dr. David Turner and Dr. George Georgiou, for their suggestions and help.

Specifically, I would like to acknowledge my two teammates Darrion DeMelo and Joriz DeGuzman in the development and debugging of this project.

Also, I would like to extend my thanks to many other friends and family who helped me during my master program.

In the end, the support of the National Science Foundation under award #9810708 is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 History

RMT (Recursive Multi-Threaded) Software Life Cycle was developed by a master's thesis [3] (Scott Simon) in 1997. RMT is a way to view the software process through threads of the software life cycle where each member of the organization is assigned a thread. In 1998, Chung-Ping Lin [4] developed a software process management tool based on RMT. The tool is used for monitoring and predicting software development progress. In 2000, another student Yi-Chiun Kuo [5] added the multi-database function for the RMT in his Masters Project. The Multi-database is used to support multiple software development sites. In 2003, two undergraduate students, Darrion DeMelo and Joriz DeGuzman, under the MII software engineering research group supervised by Dr. Arturo I Concepcion re-implemented RMT by PHP and created an easier-to-use interface and even more functionalities. Because of the changes, RMT is given a new name ReMoTe (Recursively Estimating Multi-Threaded Observation Tool Enterprise).

1.2 Propose

The principal purpose of this project is to provide support for multi-database to ReMoTe. Most internationally positioned companies have several databases, each with its own branch. Each branch serves local needs unless there is a user that requests data from another branch via the Internet. The three databases used in this project are Oracle database, Microsoft Access database, and mySQL database (shown in Figure 1). Therefore, no matter where the location of each database or project team is, central management can view the information of each project or sub-project such as personal information, position, and progress of each team member who is involved can be achieved. The benefit of ReMoTe is that each database can be geographically independent of the other. ReMoTe software process management tool uses the PHP classes provided the technology to access multi-databases transparently. The multi-database system allows insertion, retrieval, deletion and update of information regarding software projects stored in ReMoTe. Moreover, the ReMoTe project also requires an interface to interact with those databases.

A scenario of how ReMoTe will be used is as follows:

Let us say a software company in L.A. subcontracted three software companies in Bombay, India, Beijing, China

and Manila, Philippines to build components of a large software project the L.A. Company is building. Each component has its own ReMoTe database where they store information of all software artifacts in connection with the software component they are building. See Figure 1.

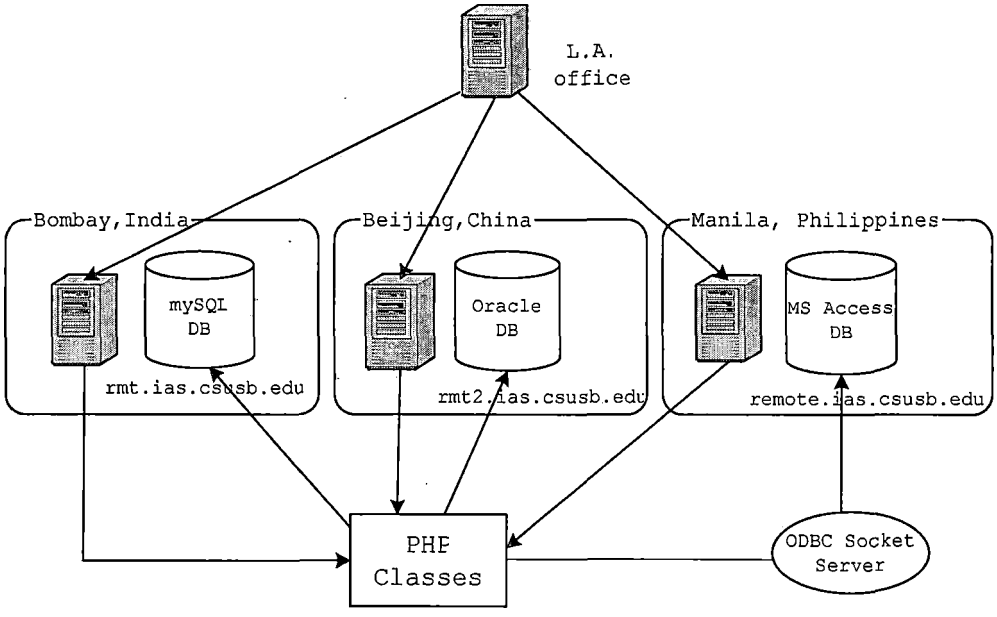

Figure 1. ReMoTe Multi-database Support

Each company may use different databases, as shown in Figure 1, and they can access their own local information on ReMoTe. If the L.A. office wants to determine the overall progress of the entire project, which includes all the

subcomponents being built by the different companies, this can be done through the ReMoTe multi-database support. Information from the different databases will be transparently sent to the L.A. office and ReMoTe will present to central management the overall progress of the entire project.

CHAPTER TWO

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Software Interfaces

The Software Requirements Specification (SRS) clearly defines the improvements and additional functionalities to the RMT (Recursive Multi-threaded) software process management tool. The product will now be called ReMoTe (Recursively Estimating Multi-Threaded Observation Tool Enterprise) primarily targets a specialized technical audience, and this specification is presented in a form suitable for technical persons, managers and software engineers.

This SRS provides the following benefits:

- Provides a basis for requirement and resource estimation.
- Provides a baseline for validation and verification.
- Serves as a basis for enhancements and additional features.

This document is intended for the following persons and organizations:

- ReMoTe Tool product management.
- All design and development personnel on the ReMoTe tool project.

- ReMoTe tool QA/Test personnel.

2.1.1 Scope and Limitations

ReMoTe is the improved and the commercialized version of RMT. The ReMoTe Software Process Management Tool offers the following functionality:

- CVS support.
- Allow easy observation through all projects currently stored in the database.
- Allow viewing both personal and overall progress, tracking of all users' activities.
- Maintain historical logging of all project data
- Multiple database support.

The ReMoTe Project Management Tool offers the following limitations:

- Any project team member can look at their entire project that they are assigned to but not the project of other teams. Management.
- Anyone can leave a message at the message board, but no one can edit or delete it except the administrator.

2.1.2 Definition of Terms

This section defines terms and abbreviations used in the ReMoTe project document.

- Apache

Apache is an open source HTTP server for UNIX, Windows NT, and other platforms.

- Browser

A program which allows a person to read hypertext. The browser gives some means of viewing the contents of nodes (or "pages") and of navigating from one node to another.

- CVS - Concurrent Versions System

CVS is a source control tool which allows multiple people to simultaneously view and edit code. CVS keeps a history of all changes that have been make to the code, along with who make the change and when it was committed into the repository.

- DAO - Data Access Objects

The DAO is a group of object which implements the access mechanism required to work with the data source.

- ER diagram

Diagrams that use Entity-Relationship model to design or describe database.

- GUI - Graphical user interface

A user interface based on graphics (icons and pictures and menus) instead of text. User enters data using both a mouse and keyboard.

- HTML - Hyper Text Markup Language

HTML is the lingua franca for publishing hypertext on

the World Wide Web. It is a non-proprietary format based upon SGML, and can be created and processed by a wide range of tools, from simple plain text editors.

- Hyperlink

A link from a hypertext files to another location or file; typically activated by clicking on a highlighted word or icon at a particular location on the screen.

- Iteration

Iteration is the repetition of a process. It describes a specific form of repetition with a mutable state. It also can be considered as a different version of a project.

- Multi-Database

A multi-database system is an environment which data is stored in two or more database instances are accessible as though these data were in a single instance.

- ODBC

A standard for accessing different database systems. The goal of ODBC is to make it possible to access data from any application, regardless of which database management system is handling the data.

- ODBC Socket Server

ODBC Socket Server is an open source database access tool that exposes Windows ODBC data sources with an XML-

based TCP/IP interface. ODBC Socket Server runs as a Windows NT/2K Service. The server was coded using MS Visual C++ 6.0 and uses standard system calls; the client was coded in PHP. All source and project files are provided with the distribution.

- OOP

Object - oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic.

- PHP

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

- Prototype

An original, full-scale, and usually working model of a new product or new version of an existing product.

- ReMoTe - Recursively Estimating Multi-threaded Observation Technology Enterprise

ReMoTe Utilizes the RMT software life-cycle that supports the monitoring of progress during development and addresses the specific needs of the developing object-oriented software.

- RMT - Recursive Multi-Threaded

Scott Simon developed RMT software life cycle model as his master thesis at California State University, San Bernardino. RMT is designed to monitor progress and organize development using a thread-based approach. RMT is an iterative life cycle tool that supports incremental and parallel development.

- SQL

SQL is abbreviation for "Structured Query Language". It is used for requesting information from a database.

- Threads

The term thread is a section of code executed independently of other threads of control within a single program. Specific in ReMoTe, it returns a person's individual project(s) (artifact(s)) for a given iteration for a prototype.

- UML - Unified Modeling Language

The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system.

2.1.3 Organization

The Masters Project document is divided into six chapters. Chapter one introduces the history and goal of ReMoTe. Chapter two provides the software requirements specification, which specifies the functionalities of ReMoTe

that will be implemented in this Masters Project. Chapter Three consists of the software design. Chapter Four documents the software quality assurance process. Chapter Five presents the maintenance required from the project. Chapter Six presents the conclusions drawn from the development of the project and future directions. The Appendices containing the project follows Chapter Seven. Finally, the references for the project are presented.

2.2 Overall Description

2.2.1 ReMoTe Overview

The users of ReMoTe are most likely to be technical or semi-technical (QA staff, project manager) individuals familiar with the field of computing. As such, the user knows technical aspects of the ReMoTe system, and can obtain maximum functionality from it.

The ReMoTe system is a Web-based application that allows developers, product managers and others involved in the development process to assess the current status of a sub-project or the entire project. The ReMoTe tool will run on any platform containing browsers that support HTML V.4.0 and JavaScript. As a Web-based application, interface elements such as edit fields, combo boxes, and other common

windows GUI controls will retain the look and feel of their parent platform.

ReMoTe interfaces are based on HTML V.4.0 and client side JavaScript supported by the client browser. Because of this, ReMoTe does not require any functionality that relies on client-side executable code outside of the browser environment.

2.2.2 Product Perspective

System Interface ReMoTe has three-tier architecture. The first tier is the Client that displays user interfaces in Web browsers (preferred Internet Explorer 5.0 or higher version). The middle tier is the Web Server that uses Apache that also implements JavaScript. The third tier is the Database Server using mySQL, Oracle and MS Access. The Web Server communicates with the Client using https and with the Database Server via PHP and ODBC Socket Server. Figure 2 shows the hardware and software deployment of the whole project from L.A., Beijing and Manila as examples.

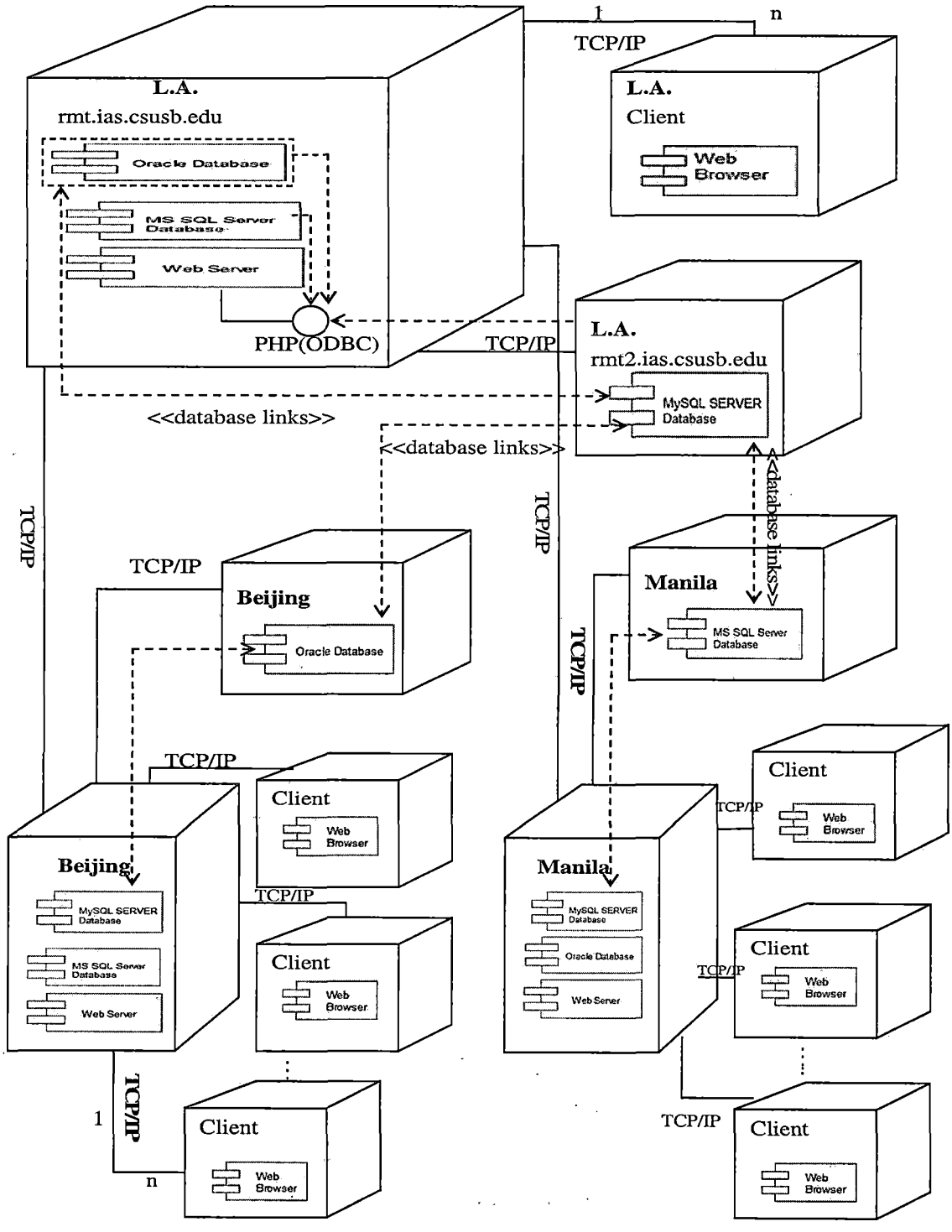

Figure 2. Deployment Diagram

User Interface The user interfaces for the multi-database project are PHP classes that are written in object-oriented approach. Without multi-database, ReMoTe software process management tool can only run on single server and use only one centralize database. Working with this multi-database, ReMoTe not only can run on different servers but also can be accessed by different software development teams located in geographically different locations.

The following sections detail the user interfaces:

- ReMoTe User Login Page

This page is the starting page for the system. The user will enter a Login ID and Password assigned by a user with account managing privileges. There are two buttons. The Login button allows the user to attempt a login. The Reset button clears the Login ID and Password fields.

- Prototype Page

In this page, the user has to supply a prototype name. After submitting the prototype name, the user can add links to this thread by clicking on the "Manage Thread" link.

- View Project Page

This page shows the whole project organization, including all the teams.

- Add Message Page

This page allows the user to create, send, and receive messages.

- View Message Page

This page shows all the messages on the message board and allows the user to reply to any of the messages.

- Add Manage Thread Page

This page will allow the user to add threads to this prototype.

- Set Team Name Page

This page allows the manager or a team leader to create team names for this prototype.

- Set Delivery Dates Page

This page allows the manager to set delivery date and phase name for this prototype.

- View Progress Page

This page allows the user to view the overall progress using bar graph.

- Using Gantt chart to View Stats Page

This page allows the user to view the duration of the tasks against the progression of time using Gantt chart.

- Admin Login Page

This page allows the systems administrator to login to the administration page.

- Admin Accept User Page

This page allows the systems administrator to accept, delete and edit users. Also, the project manager has to be created by the systems administrator in this page.

- Edit Thread Page

This page allows the manager to edit threads.

- View Person Page

This page provides viewing of personal pages. There are five choices for the user on this page. The user can choose to view any project, to set up which project to view, view the progress, add messages, and view messages.

- Set Projects to View Page

This page allows the user to set up the project's information.

- View Project Page

This page shows every team member's information and project progress of the prototype which the project is setup.

Hardware Interfaces. The server operating system manages hardware interface issues. There is no hardware interface needed in ReMoTe.

Software Interfaces. This project uses mySQL database, Oracle database, Access database, Apache HTTP server, and PHP classes. Therefore the server side portion of this project requires a platform that offers a Web server, PHP

runner, Oracle database, MS Access database and ODBC Socket Server. The current iteration of ReMoTe is designed to run on the Linux platform using Apache PHP engine. Oracle database and MS Access database run on Windows platform.

Communication Interfaces. ReMoTe relies on the client operating system, and the server PHP engine and operating system to manage communication issues. The system can be executed in Internet Explorer, Firefox and Netscape which are the most common browsers today. In addition, the other communication interface between database and the software product goes through ODBC.

Memory and Hardware Constraints. There is no ReMoTe client-side memory constraint issue.

Additionally, the server requires the following components for optimal content generation in a multi-user environment.

For the server with mySQL database:

- 256 MB or greater memory size.
- Fast architecture machine, something along the lines of a PII 500 or greater.

For the server with Oracle 9i database:

- 512 MB or greater memory size.
- Disk space equal to the system's physical memory, or 1 GB, whichever is greater.

Adaptation Requirements. This project has no site adaptation at this time.

2.2.3 Product Functions

There are two main functionalities provided by this project. The first is to allow ReMoTe users to create, delete, update and retrieve projects, threads, personnel and iteration information from the database. The second is to support multiple sites access showing only a single ReMoTe hierarchical tree. In other words, a ReMoTe project can store its information of the sub-components or a large software project in different databases at different locations. Once a ReMoTe user wants to access data that is not stored in his/her own local database, ReMoTe can still access the desired information from the appropriate site through multi-database support. Figure 3 shows the use case for four types of users: software engineer, project manager, administrator and management. The regular user can do everything the project manager does except the setting action. Setting action includes setting iteration number, messages and Bugzilla location. Administrator is the only person who has the authorization to create the project manager and delete messages. Management is the view only person who can select which project to view.

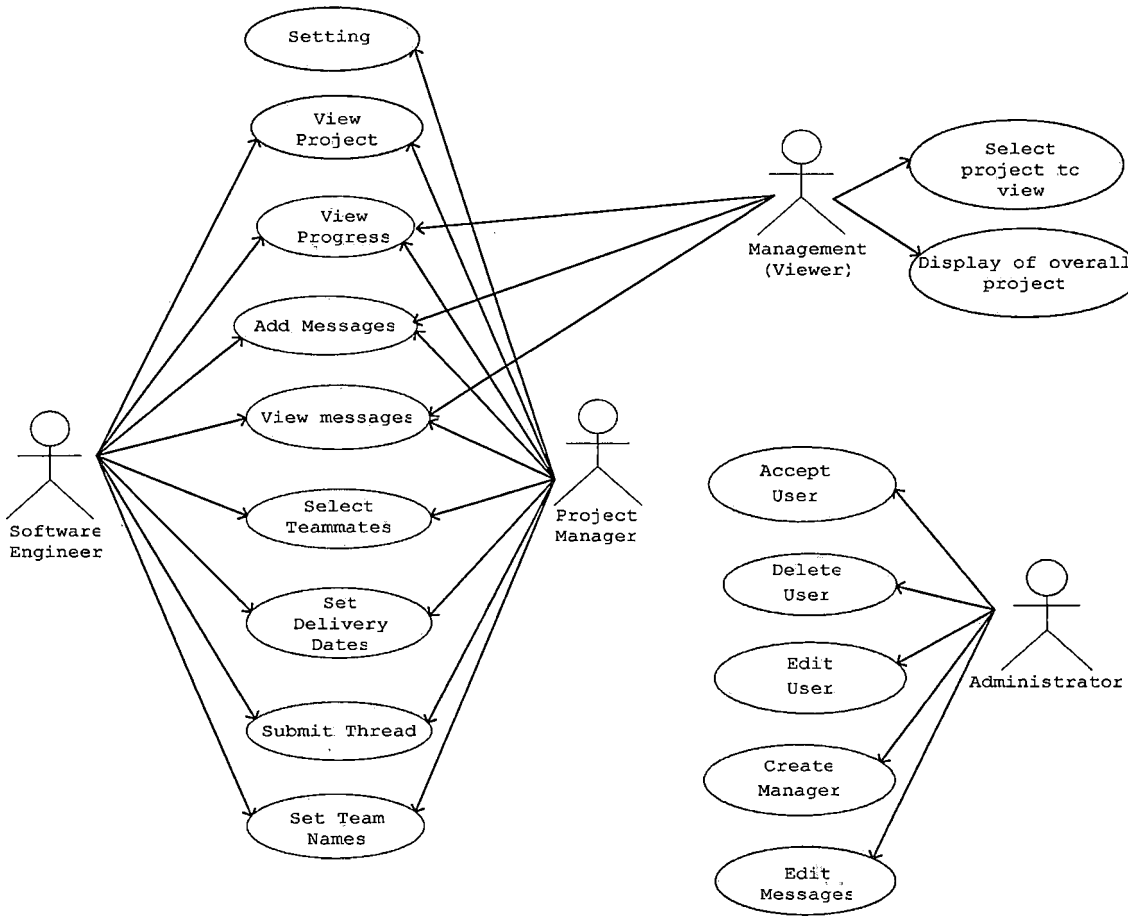

Figure 3. Use Case Diagram

2.2.4 User Characteristics

This project is primarily designed for the software developer. There are only one or two users in the management group. This user is the vice-president, president of the company, or their representation. The management user will have all rights and abilities to view the overall progress.

The system Administrator in the company will use the tool in order to set up the Web server and client accounts. The user in the client group must be a software engineer. The user will use the interface of Web pages to log in and then input the threads which he/she has finished.

2.2.5 Constraints

The server relies on Apache, PHP, ODBC Socket Server, Oracle and mySQL. While many platforms offer these functionalities, this project is currently developed for the Linux platform. As such, certain additional work may be necessary to port this project server to other platforms or other databases.

2.2.6 Dependencies

This project has the following dependencies:

- The same relation (table) types exist for all the databases.
- One server platform is a Linux system with the following components installed and working properly:
  1. Apache 2.0 or other web server system
  2. mySQL
  3. Redhat Package manager (rpm) or similar installation system
- One server with Oracle database.
- Another server with Windows platform with MS Access.

2.3 Specific Requirements

2.3.1 External Interface

Figures 4 to 21 show the user interface of the ReMoTe project. ReMoTe uses these interfaces to get information from the user or displays the information using the interface. The ReMoTe project has to use PHP classes to store or retrieve information from the database.

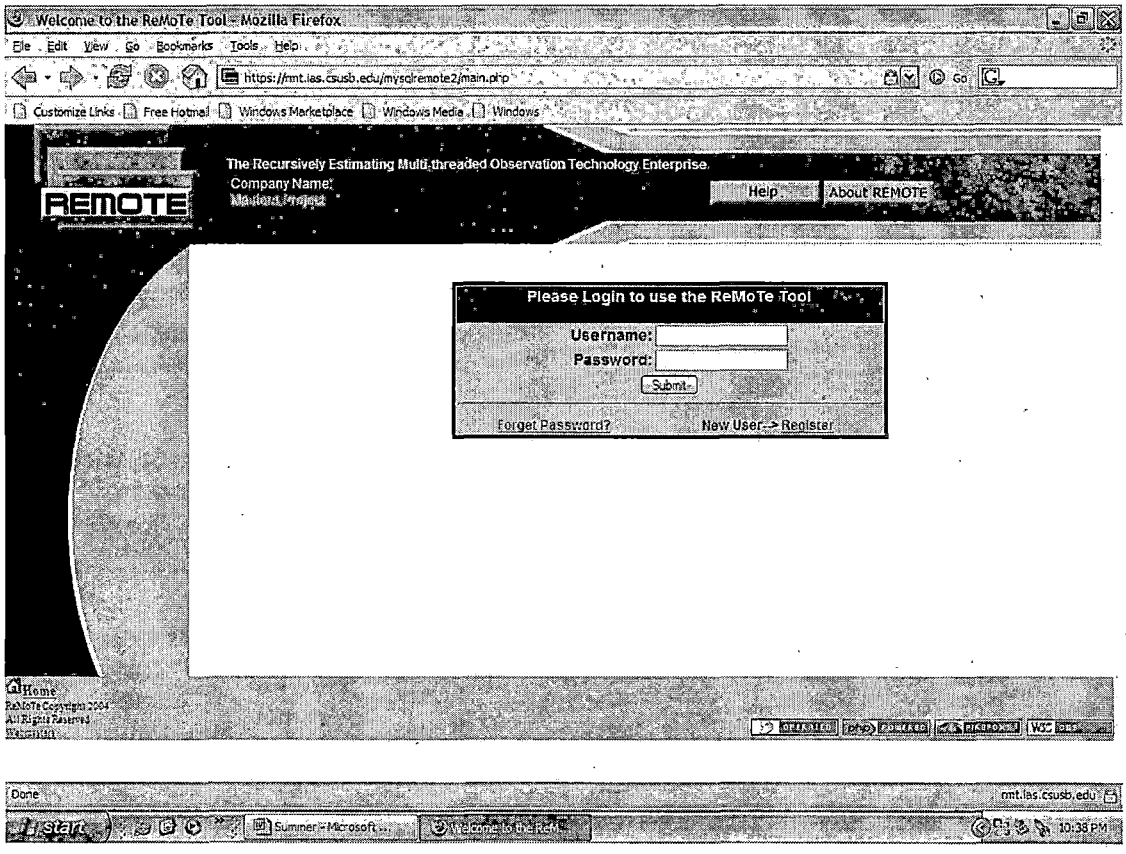

Figure 4. Login Screen of the ReMoTe

Figure 4 shows the starting page for ReMoTe. The user enters the user name and password assigned by another user with account managing privileges. There is one button that allows the user to attempt a login and two buttons respectively allow a new user to register and helps an exist user to get the password which he/she forgot. There are two kinds of users that have privileges:

1. User

2. Management

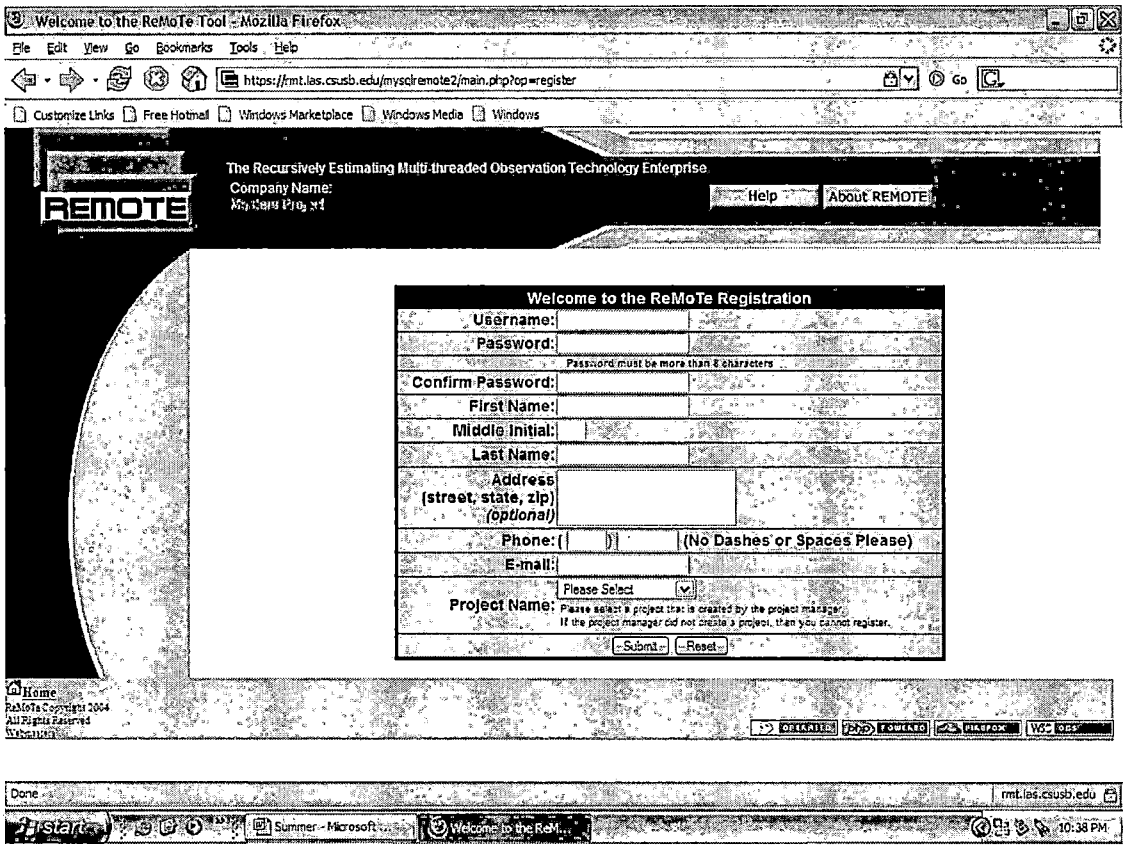

Figure 5. Registration Page

Figure 5 shows the registration page for a new user to register. There are nine items need the user to fill in. One of them is the address which is optional. And the last one item needs the user to select the project name which is created by the project manager. There are two additional

buttons:

1. Submit (enters all selections)
2. Reset (clears all previous input)

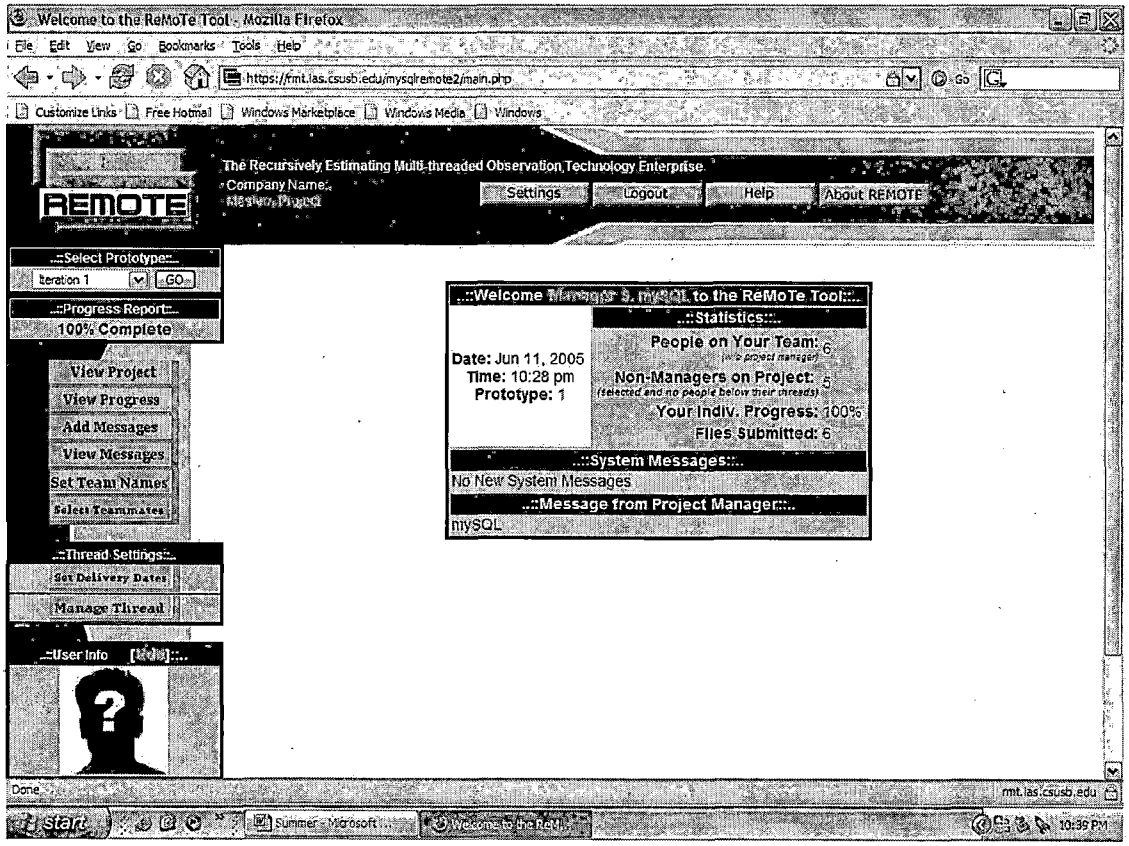

Figure 6. Introduction Page

Figure 6 shows the introduction page. The user has to select a prototype before other actions. In this page it shows the current date and time, number of team, progress,

number of submitted files, message from the project manager, and message from the system. On the top of this page there are four buttons to be chosen for the project manager. They are:

1. Settings (allows the project manager to set up iteration number, message and Bugzilla location)
2. Logout
3. Help
4. About ReMoTe

There are four categories that can be chosen on the left side of the page. They are:

1. Select Prototype

   There are two buttons in this category. The iteration button allows the user to select iteration. The Go button allows the user to set the iteration.

2. Progress Report

   It shows the percentage of progress of the software project. Also, there are six buttons that can be selected by the user. They are:

   - View Project
   - View Progress (view the Gantt Chat)
   - Add Messages
   - View Messages

- Set Team Names
- Select Teammates

3. Thread Settings
   - Set Delivery Dates
   - Manage Thread (set the thread)

4. User Info

In this category, the user can edit his/her information by clicking the Edit button. Also, by clicking the photo the user can set up his/her photo.

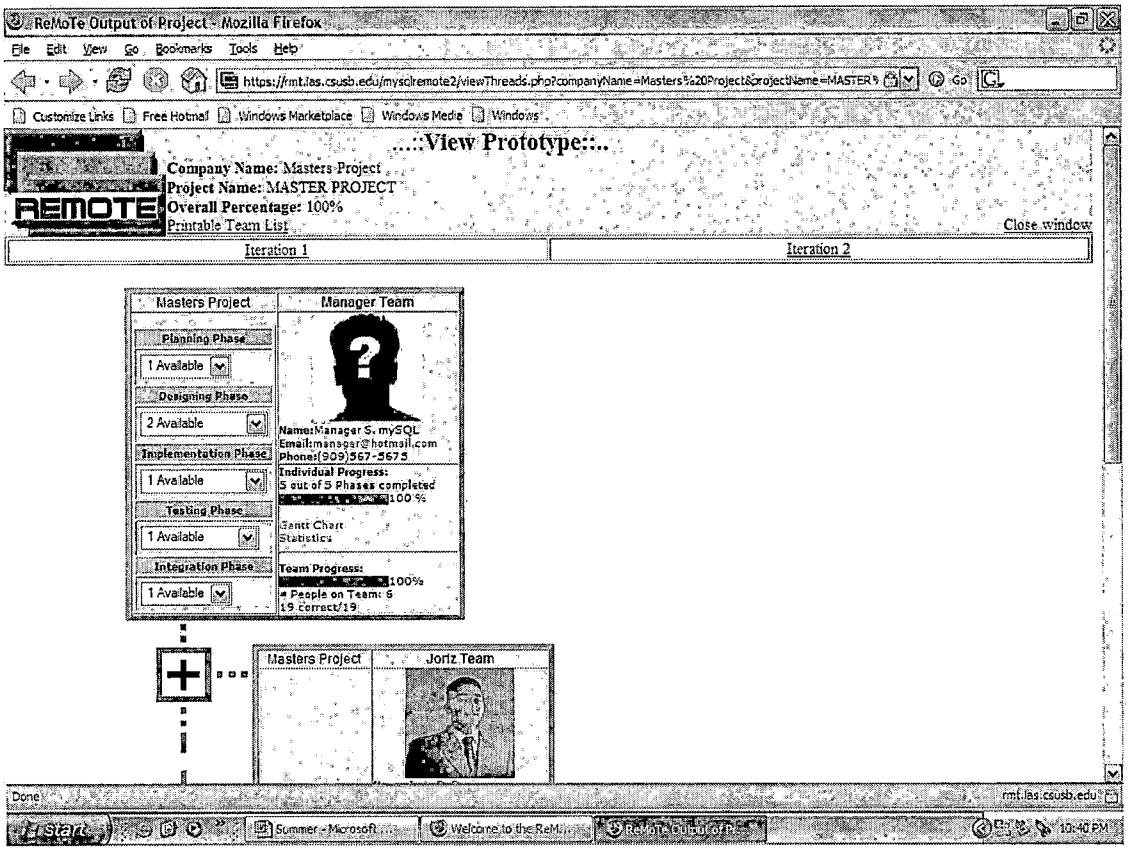

Figure 7. View Project Page

View project page shows every software developer in the project including their personal information, photo, percentage of the progress. It also shows the software development phases and in each phase there is a button that allows the user to view the thread which is submitted by a software developer. Besides, displaying a Gantt Chart, statistics can be viewed when the user clicks the buttons.

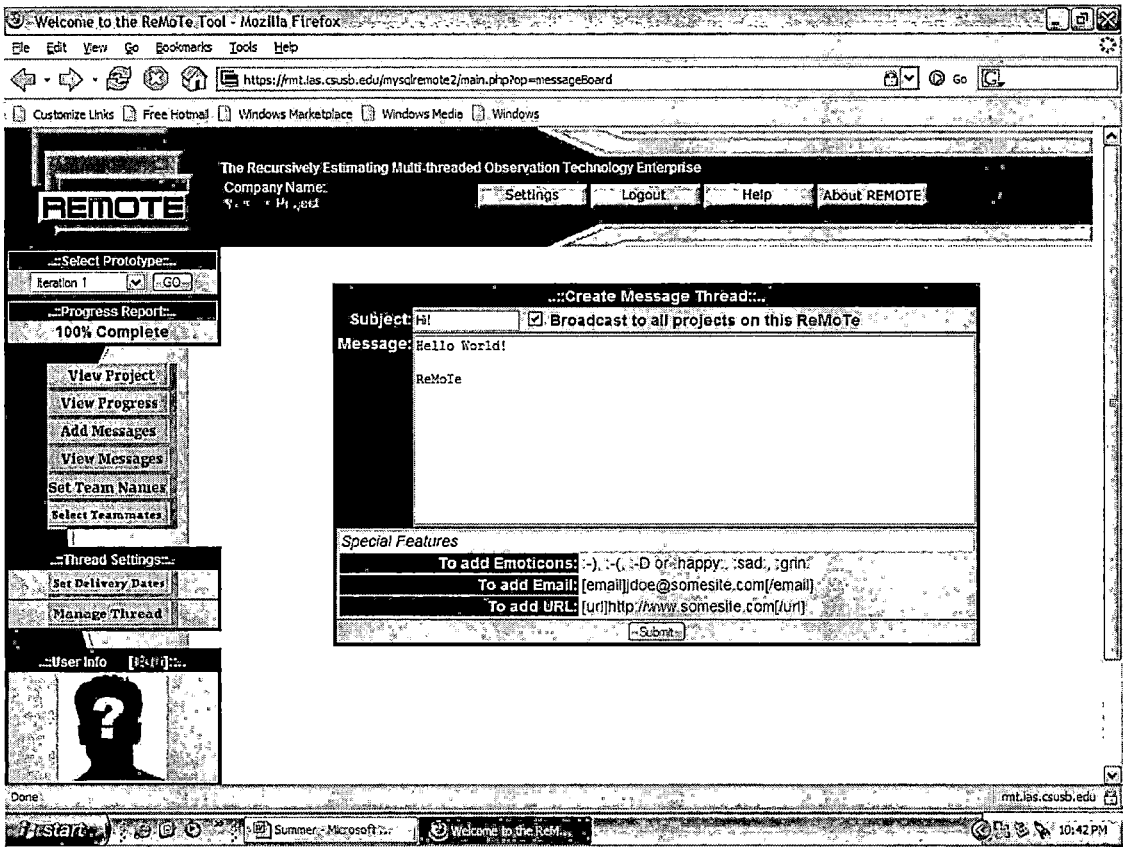

Figure 8. Add Message Page

Add message page is for the user to create messages. There is a subject and a message part. The Submit button allows the user to submit the message.

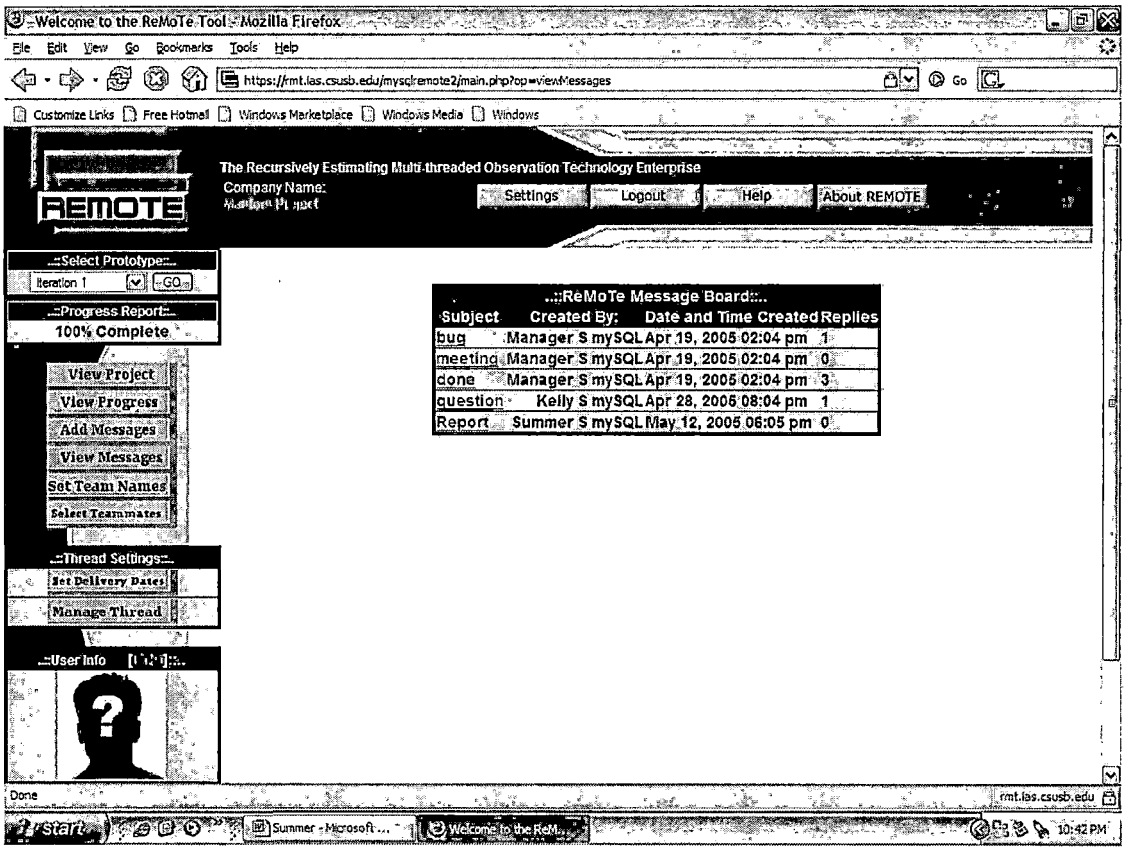

Figure 9. View Message Board Page

View message board page shows the subject of the message, date and time, created by a sender of the message and the number of replies of all the messages. The user can also click on the subject to view the content of the message.

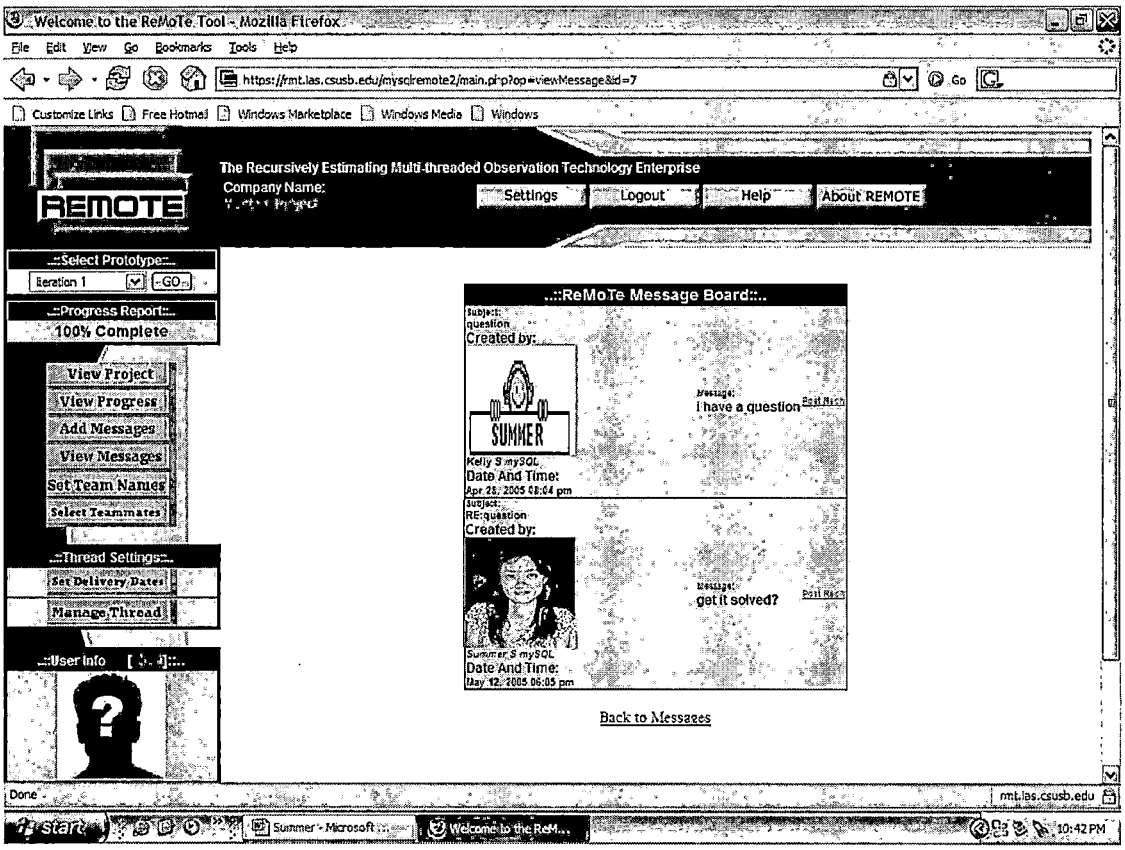

Figure 10. View Message Page

View message page shows the content of the message and all the contents of the replies for this message. For each message there is a Post Reply button to choose to reply for this message. Also there is an additional button to choose to go back to the Message board.

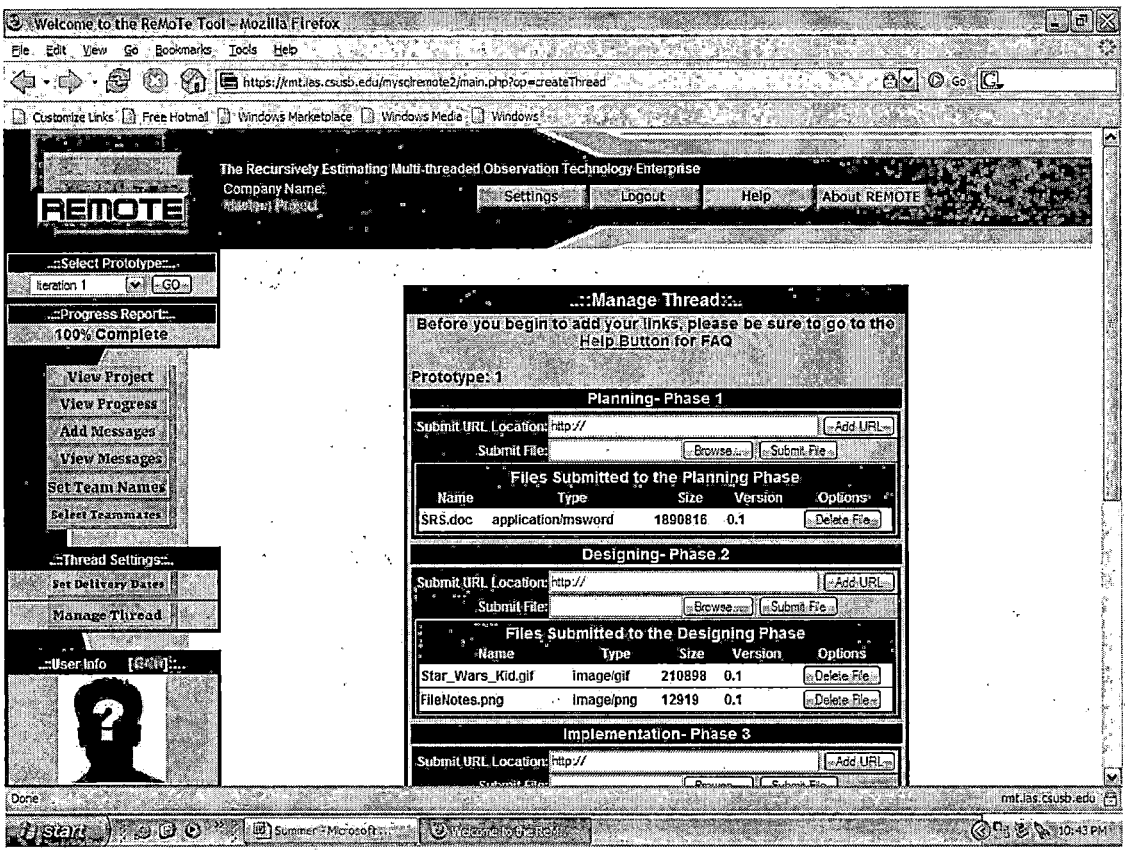

Figure 11. Add Manage Thread Page

Add manage thread page is for the user to set the file or URL that will go into each software life-cycle phase of a thread. Each phase has five buttons. They are:

- Submit URL Location
- Add URL (enter the URL thread)
- Browse (browse from the user's directory)
- Submit File

- Delete File

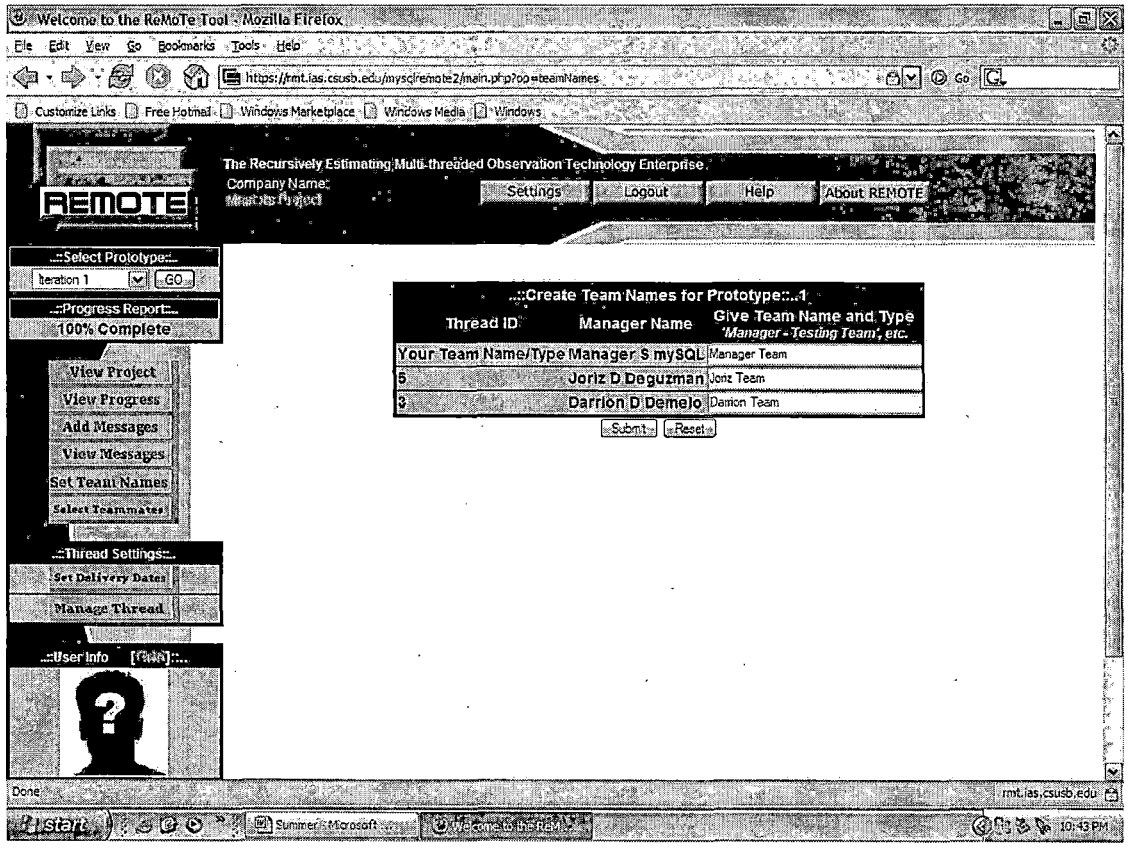

Figure 12. Set Team Name Page

Set team name page shows the team name in this project. The user can change the team name here. There are two buttons in this page. The Submit button is for the user to enter the team name. The Reset button is for the user to clear all previous input.

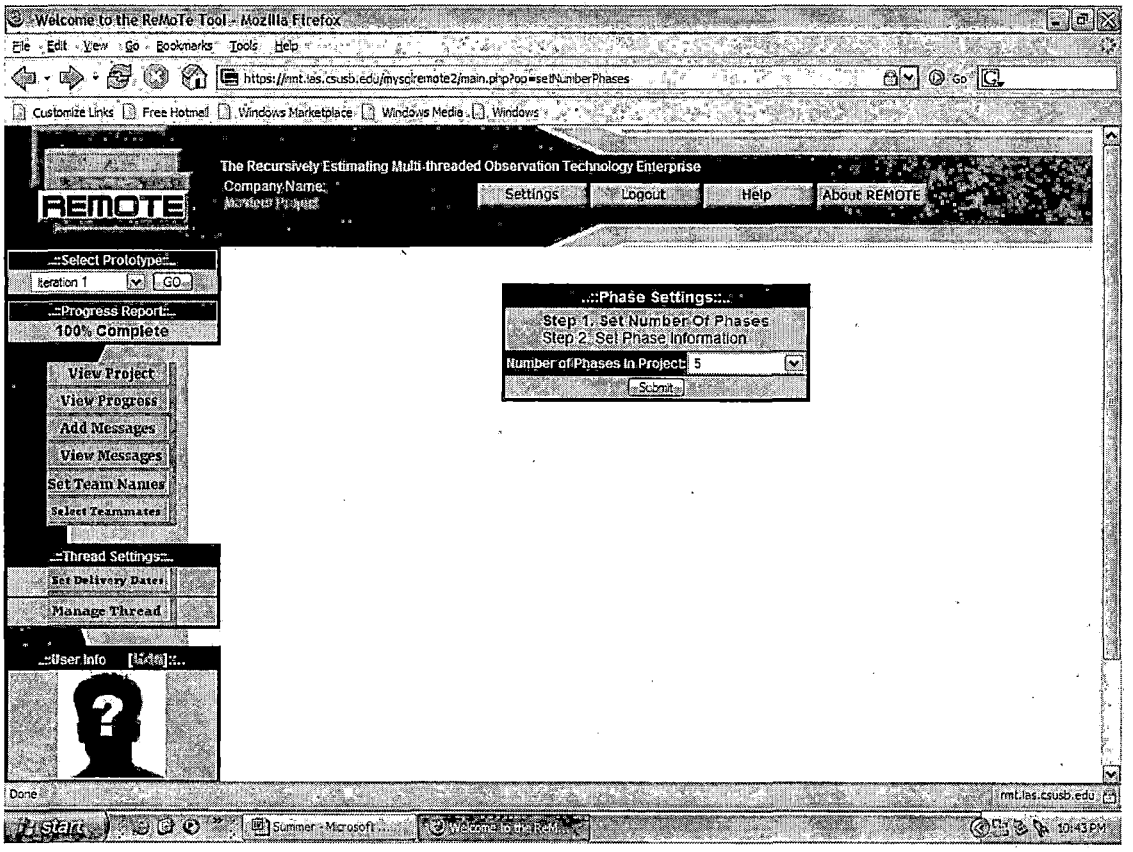

Figure 13. Set Delivery Dates Step 1 Page

This page is for the user to set the estimated delivery dates. The user has to set up the phase number first for this prototype.

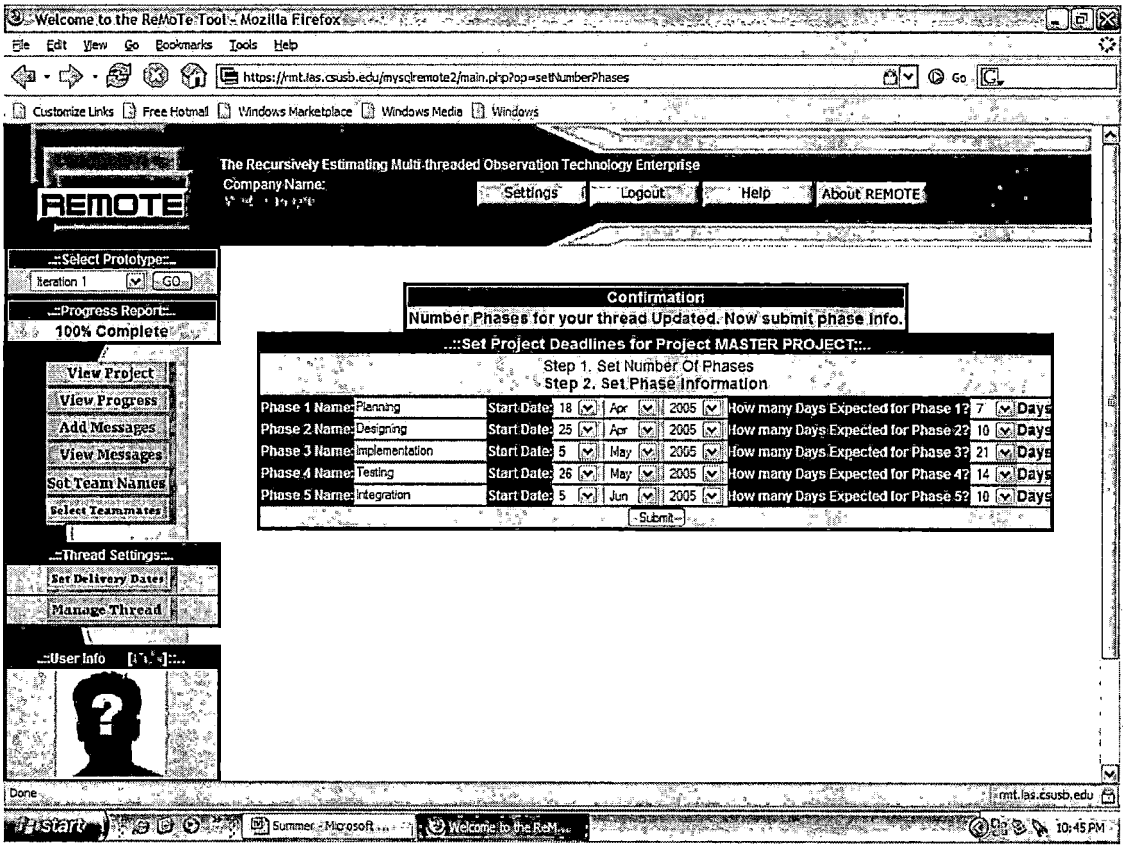

Figure 14. Set Delivery Dates Step 2 Page

This page is for the user to set the estimated delivery dates. The user has to create the phase name, enter the start date and their durations for each phases.
There is an additional button Submit that allows the user to enter all selections.

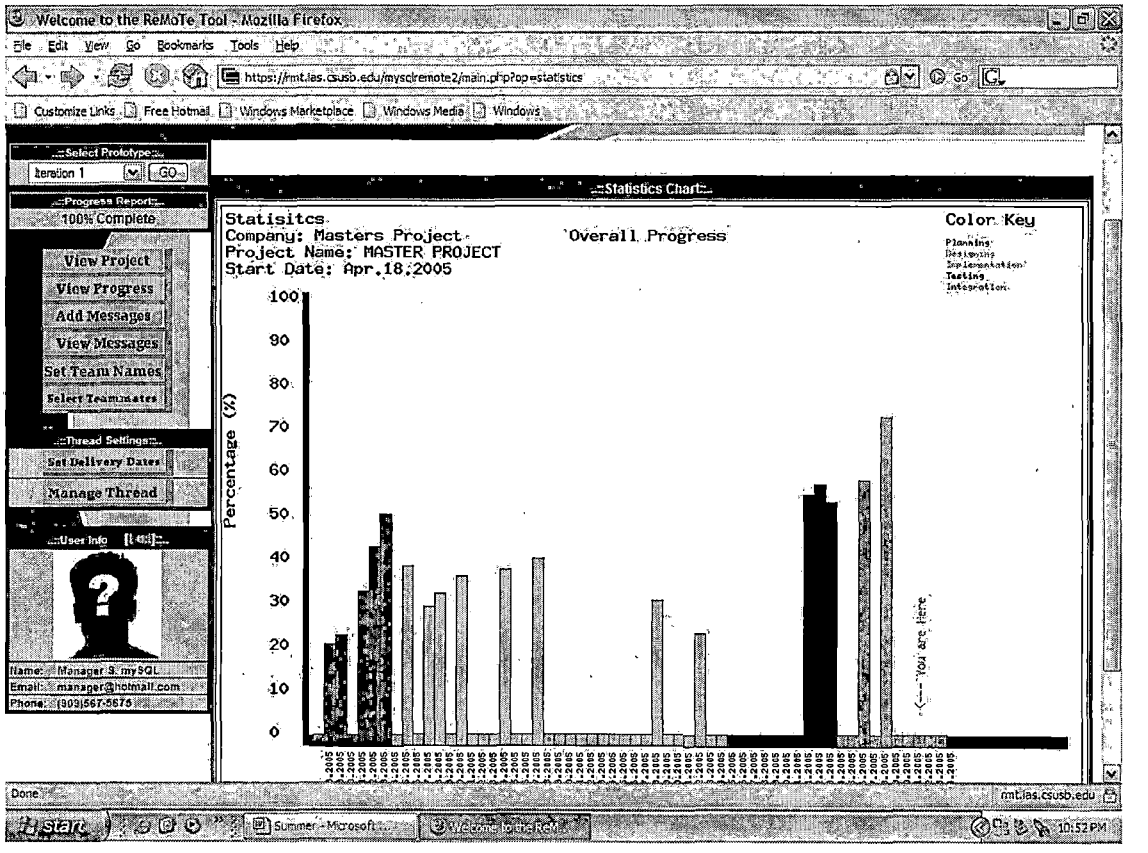

Figure 15. View Progress Page

View progress page is for viewing the percentage of progress of the user in a particular project. Each software life-cycle phase is displayed in different colors to distinguish them from each other.

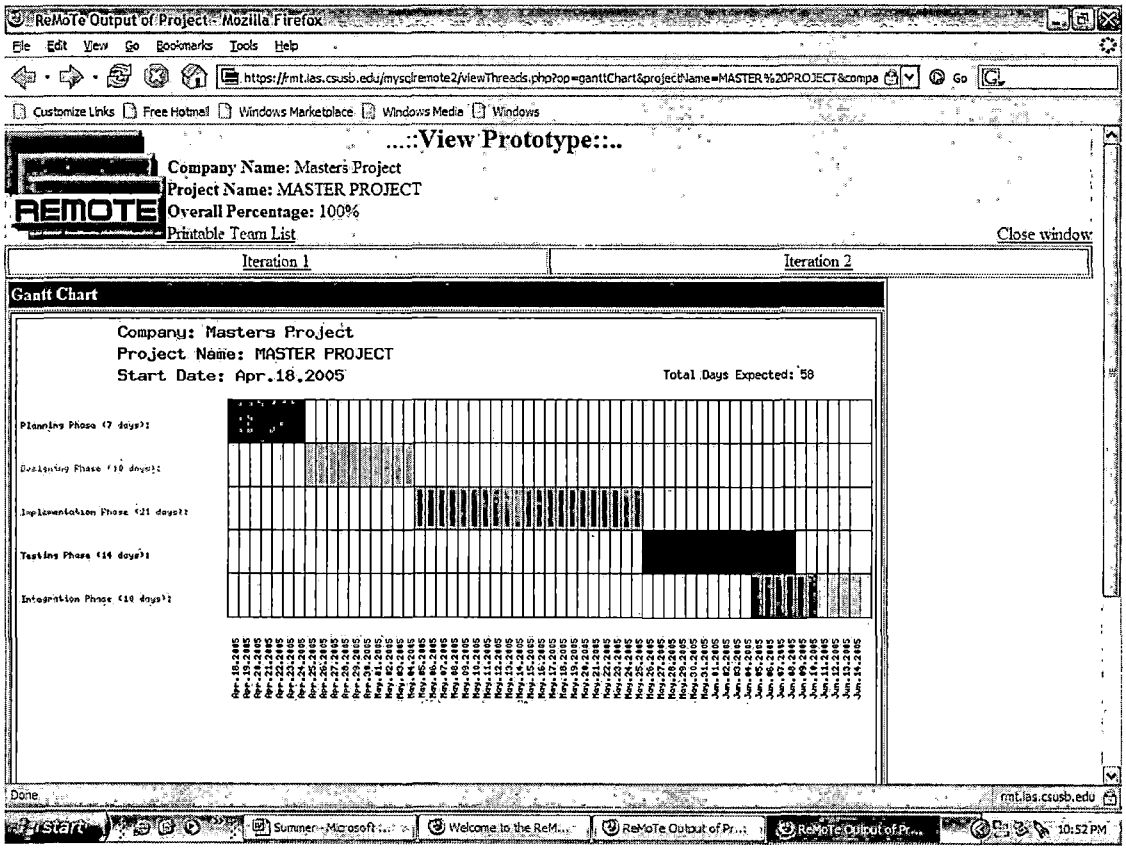

Figure 16. Gantt Chart Page

Gantt chart page displays a Gantt Chart to graphically represent the duration of tasks against the progression of time. Also, each phase is using the different colors to distinguish from each other. Before the present time the color would change to darker, after the present time the color is lighter.

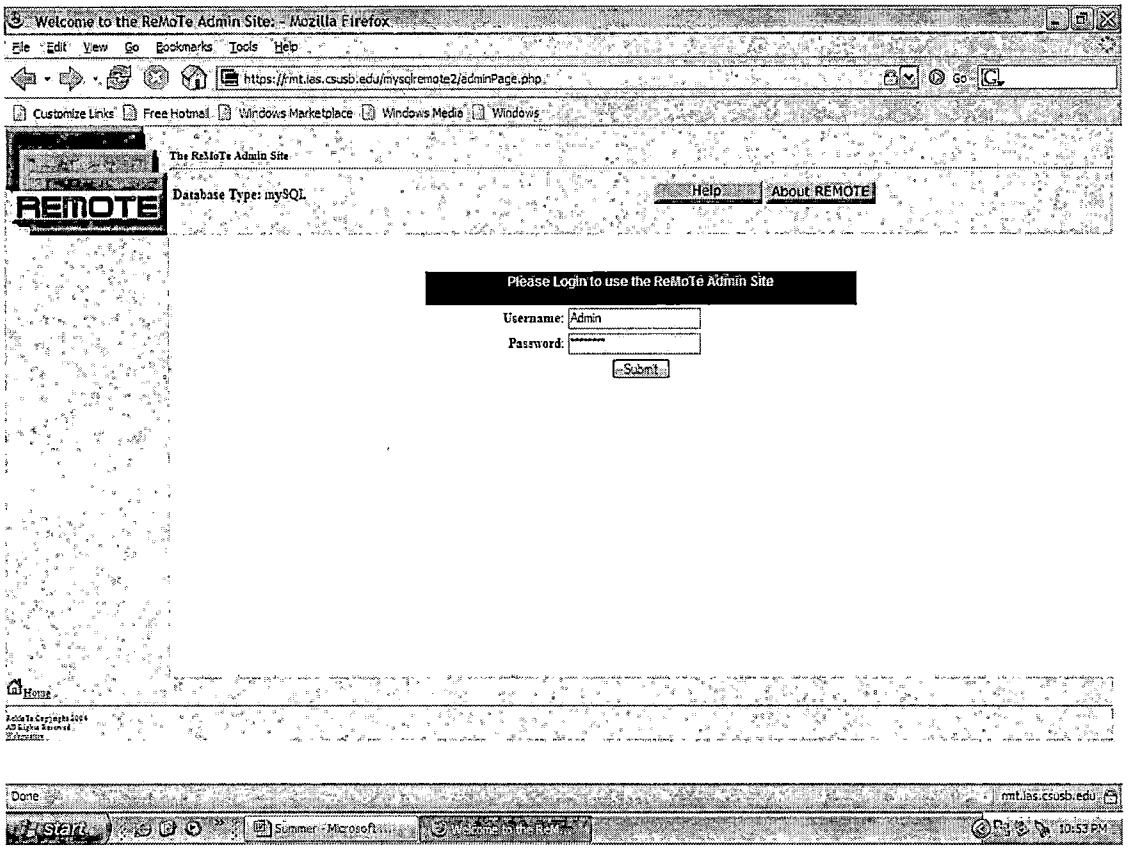

Figure 17. Admin Login Page

Admin login page is used by the system administrator to log in. The user enters the user name and password and presses the Submit button allows to attempt a login.

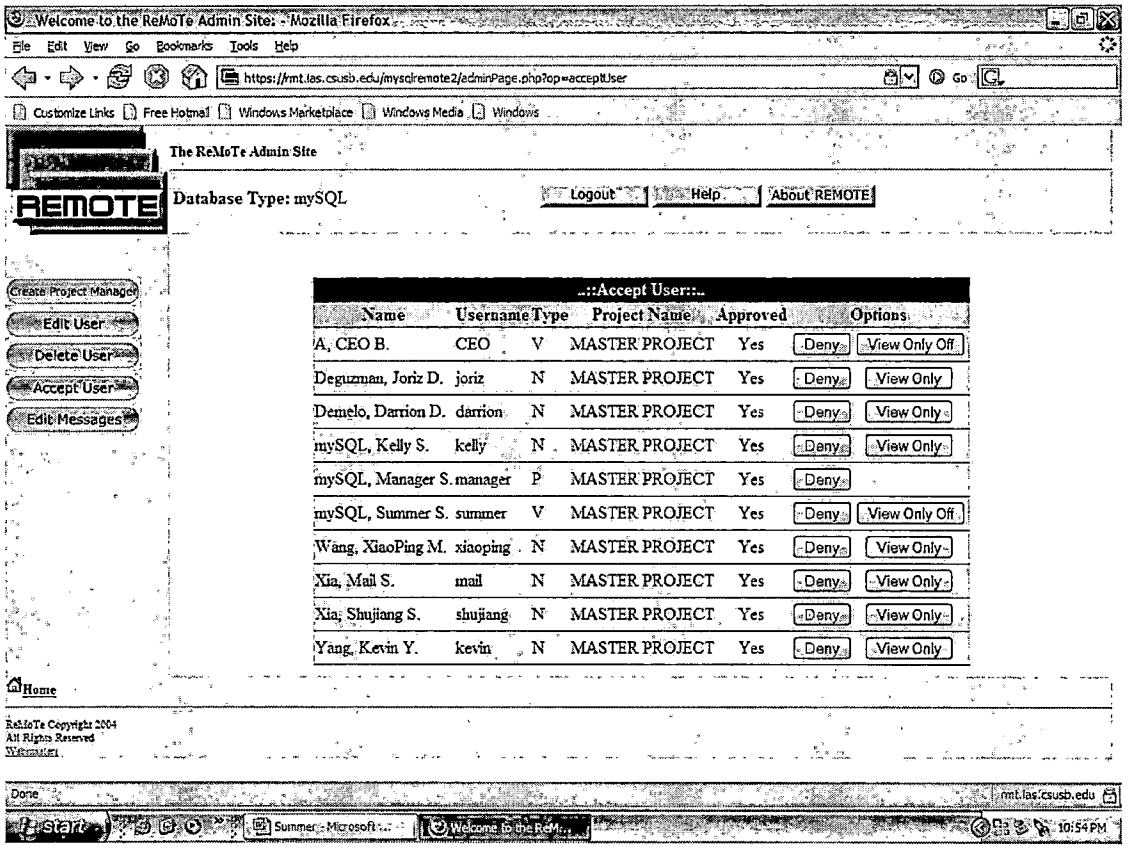

Figure 18. Admin Accept User Page

Admin accept user page is used by the system administrator to approve, delete and edit the users. Also set the user to the view only person. For each user there are two buttons, one button can be changed to Accept or Deny, the other button can be changed to View Only or View Only off. Project manager has to be created by administrator in this page. Also, in this page administrator can choose to edit the messages.

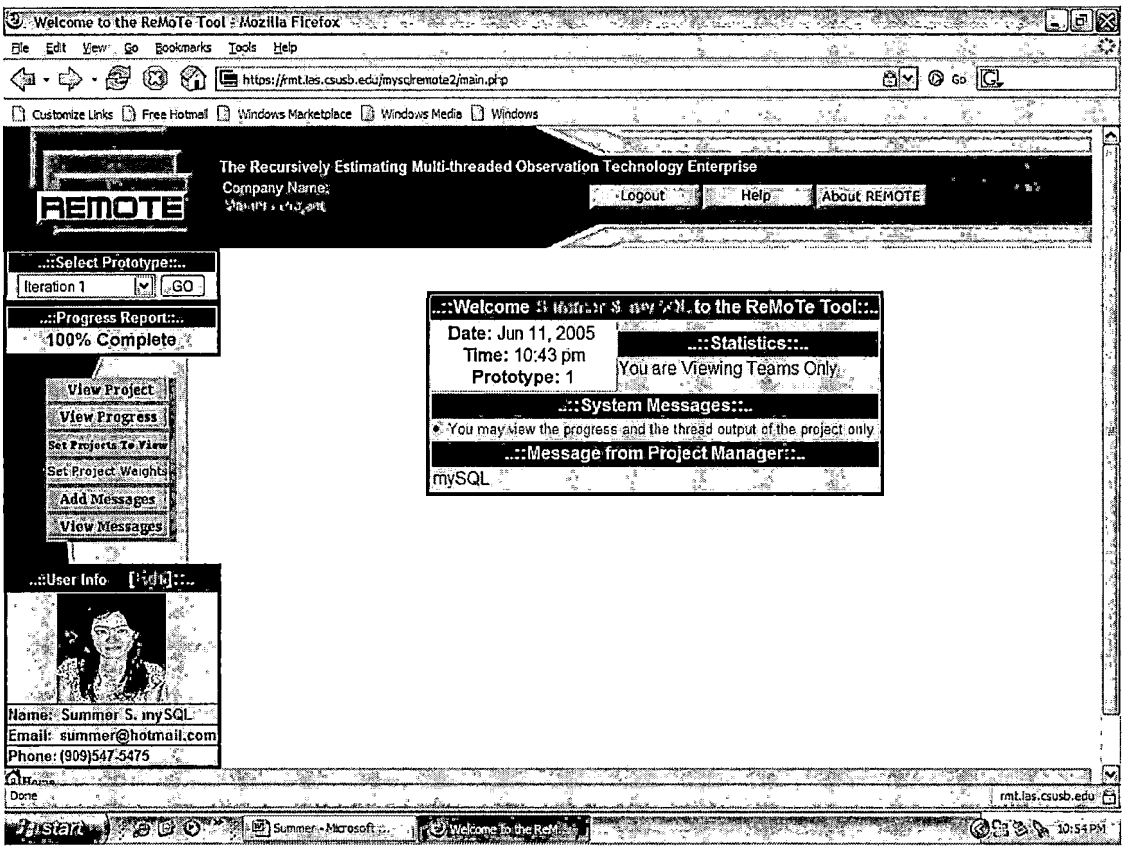

Figure 19. View Person Page

After the view only person login he/she can access view person page. In the Progress Report category there are four choices for the user. They are:

- View Project
- View Progress
- Set Project to View
- Add Messages

- View Messages

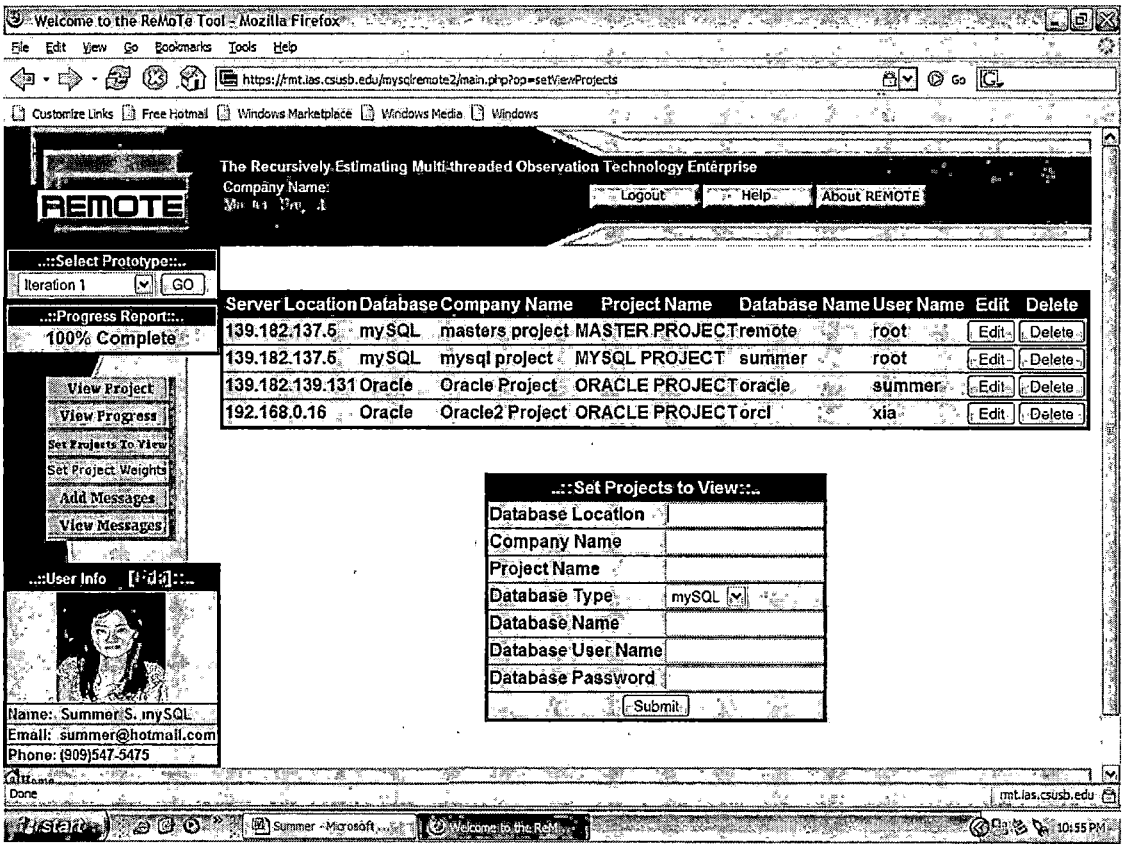

Figure 20. Set Projects to View Page

Set projects to view page is used by the view only person to set up which project to view. The user can input the project information at the bottom form and click the submit button to enter all the necessary information. The

entered information would show up on the top form. There are two buttons on the top form which are Edit and Delete. The user can use these two buttons to change the database information or delete them.

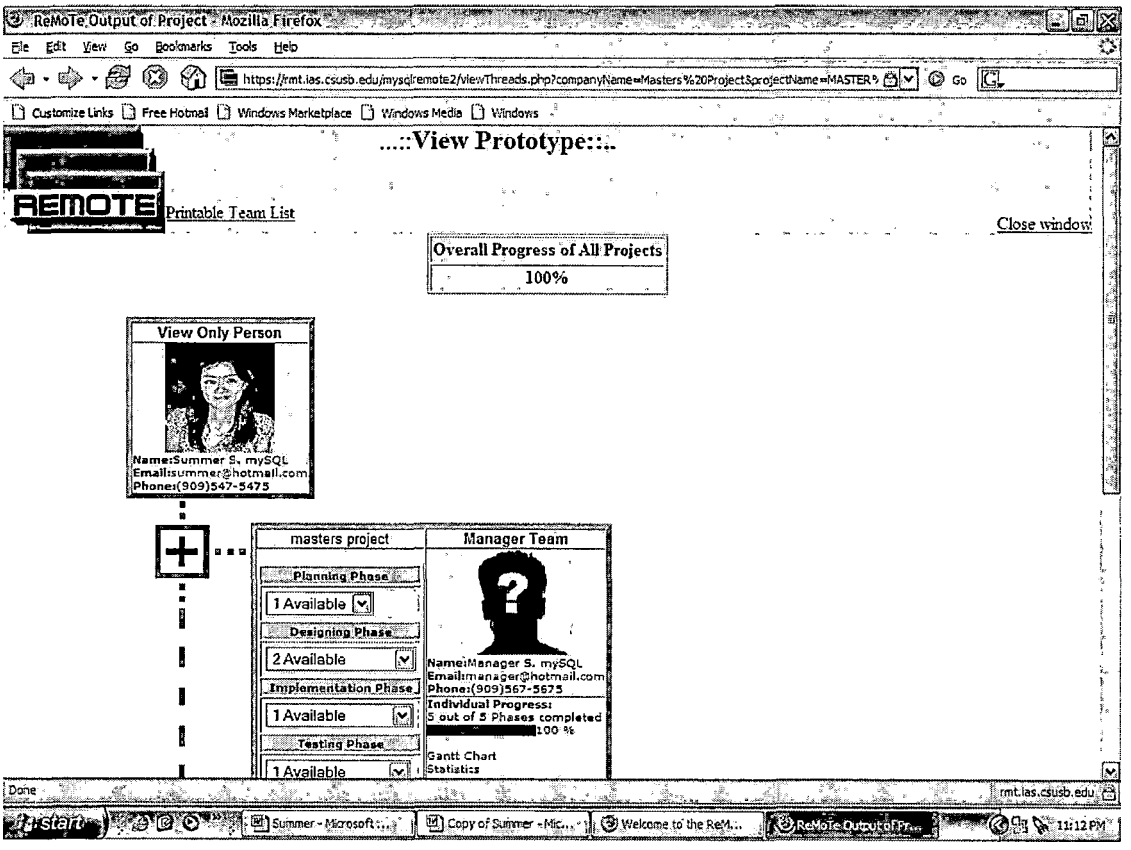

Figure 21. Thread Page

Thread page shows every software developer's information which the view only person set up.

2.3.2 Performance Requirements

The number of databases to be supported in this Masters Project is three but the number of databases that can be supported is theoretically a large number. The maximum number of simultaneous users and the data to be supported depends on the traffic on the Internet.

2.3.3 Database Requirements

There are eleven tables were created to store all the information on the ReMoTe software process management tool - they are User, Threads, Statistics, Settings, SetViewProjects, SetTeamNames, SetNumberPhases, SetDates, SelectTeam, MessageBoard and URL. Their design is shown in Chapter 3.

2.3.4 Design Constraints

There are no design constraints in this project.

2.3.5 Software Attributes

Because PHP is server side programming language, PHP supports safe programming practices on a number of levels. Web pages themselves are secured by the host machines security systems, and are reasonably safe from tampering by the general public. In Linux server, IPtable has been installed and there are two ports opened for accessing. For Windows server ZoneAlarm system has been installed and there

are two ports opened for Oracle database and accessing. ReMoTe support different user access privileges and protect the user's data.

Because PHP is a server side language and conforms to a well-defined and widely accepted database connection, they are highly portable across operating systems and across server implementations. They can be used on a Windows 2000 machine IIS Web Server and later deployed effortlessly on a high-end UNIX server running Apache.

CHAPTER THREE

PROJECT DESIGN

3.1 Architectural Design

ReMoTe used MVC architecture, i.e., Model-View-Controller. Model is an object which represents data in a software application such as a database. View is a collection of classes representing the elements in the user interface. Controller represents the classes connecting the model and the view, and is used to communicate between classes in the model and view. ReMoTe MVC Architecture shown as Figure 21.

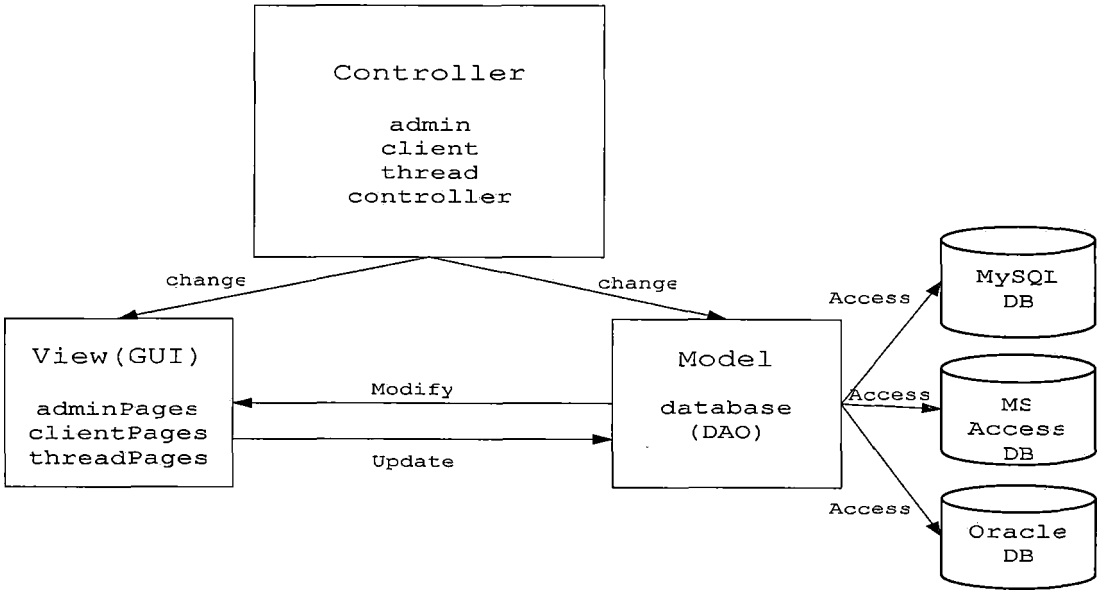

Figure 22. ReMoTe Model-View-Controller Architecture

3.2 Model Design

Figure 22 is the class diagram for the DAO classes. There are twenty-eight classes. The parent classes are Settings, MessageBaord, User, SetViewProjects, SetDates, URL, Stats, SetNumberPhases, FileNotes, AddThread, SetTeamNames and SelectTeam. All of them have set, get and the read functions. As their child classes, they respectively are SettingsDAO, MessageBaordDAO, UserDAO, SetViewProjectsDAO, SetDatesDAO, urlDAO, StatsDAO, SetNumberPhasesDAO, FileNotesDAO, AddThreadDAO, SetTeamNamesDAO and SelectTeamDAO. They make it easy to send SQL statement to relational database system and send back the result. ODBCSocketServer is the class to connect to the Access database. It includes various functions as same as other databases such as mySQL in PHP library. ConnectDAO is the core class to connect the different databases. ThreadDAO is the class to print out all the threads have been submitted by the every project number.

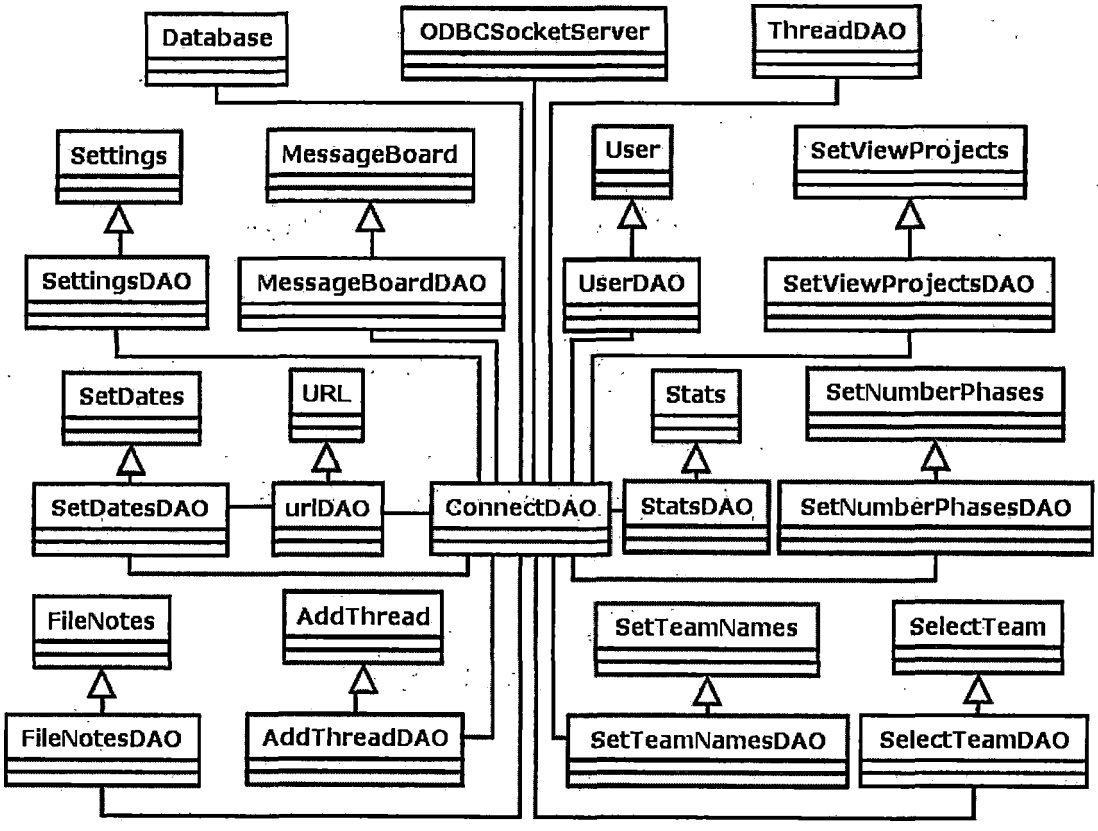

Figure 23. Class Diagram for Database Access Classes

There are eleven tables in this project. The Figure 23 shows the user table. The ID Number is the primary key for the user table. Figure 24 shows the relationship of each table for ReMoTe. As same as user table, the ID Numbers are the primary keys for those tables. All the entities and attributes are detailed in Figure 23 and 24. Tables 1 to 11 show the each table.

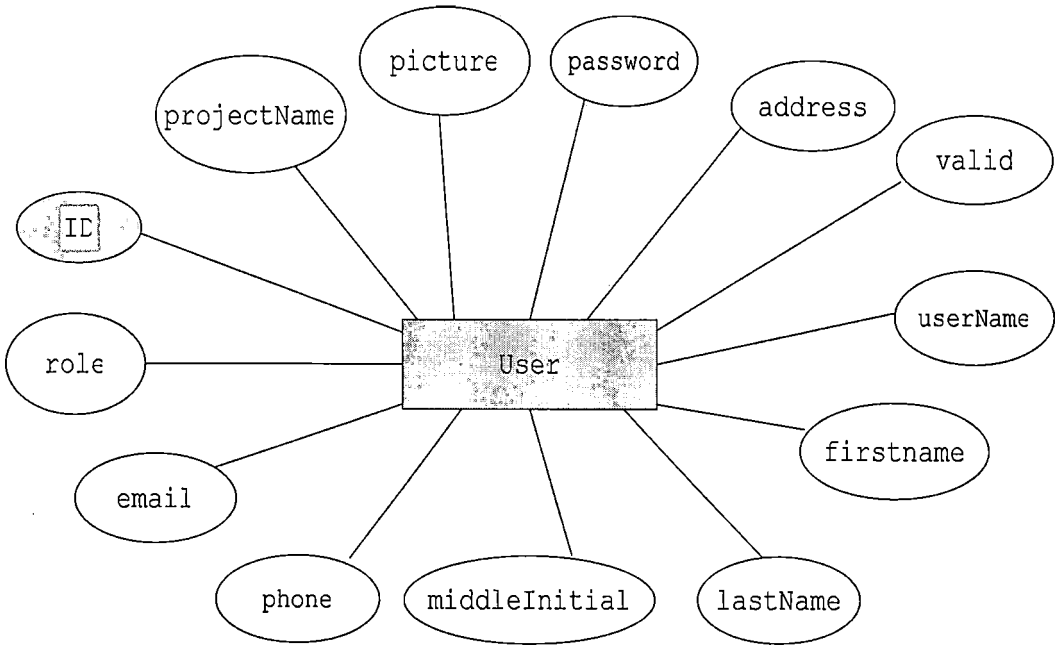

Figure 24. Entity-Relationship Diagrams for User

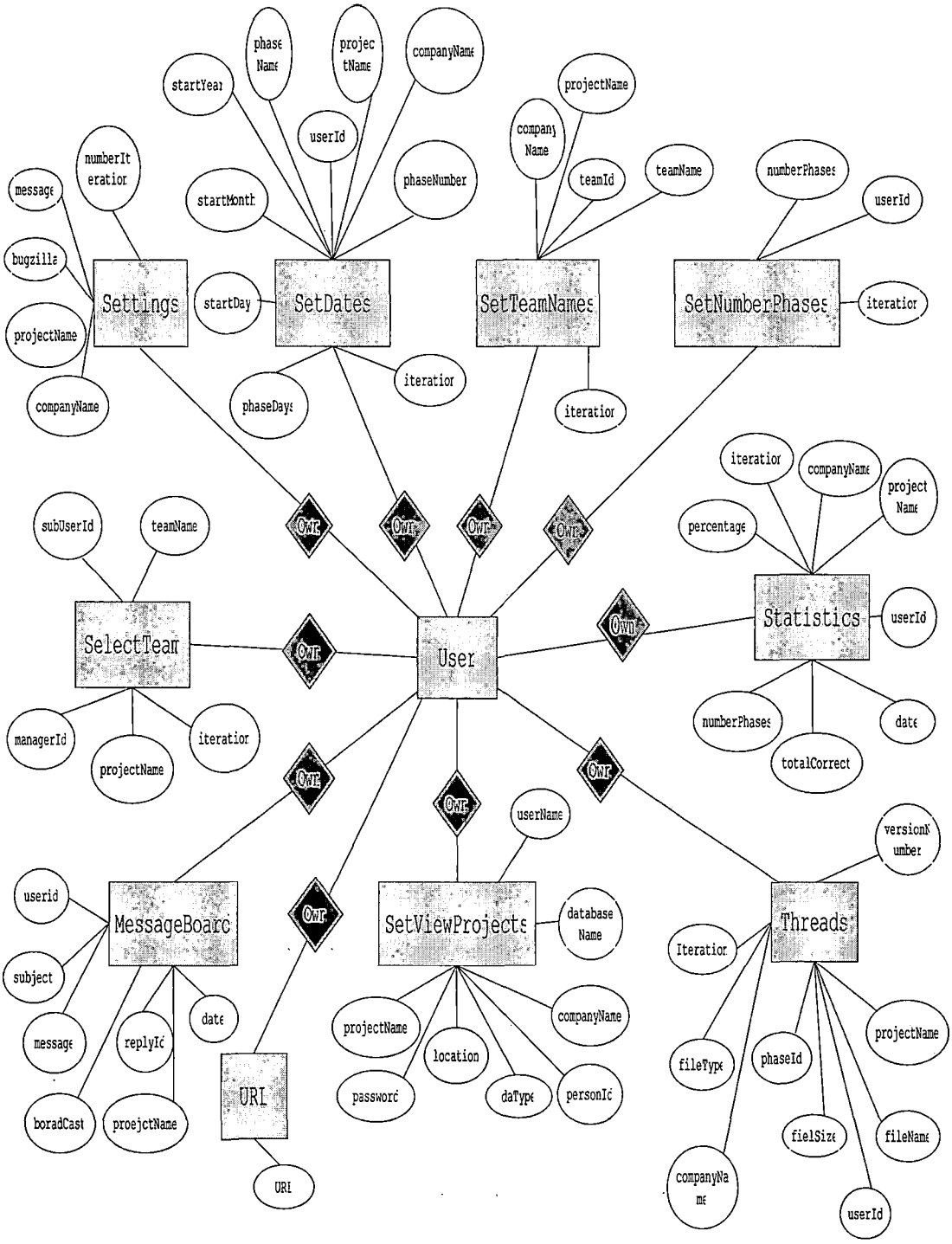

Figure 25. Entity-Relationship Diagram

Table 1. User Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
username	text	text	varchar(240)		
password	text	text	varchar(240)		
firstName	text	text	varchar(240)		
lastName	text	text	varchar(240)		
middleInitial	text	text	varchar(240)		
phone	text	text	varchar(240)		
email	text	text	varchar(240)		
role	text	text	varchar(240)		V: viewer P: project manager N: none
projectName	text	text	varchar(240)		
valid	text	text	varchar(240)		
address	text	text	varchar(240)		
picture	text	text	varchar(240)		

The user table provides the detail personal information of users.

Table 2. Threads Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
companyName	text	text	varchar(240)		
iteration	int	int	number		
projectName	text	text	varchar(240)		
filename	text	text	varchar(240)		
fileType	text	text	varchar(240)		
fileSize	int	int	number		
versionNumber	double	double	number		
userId	int	int	number		
phaseId	int	int	number		

The thread table provides the all the information of the file which the user submits.

Table 3. Statistics Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
iteration	int	int	number		
companyName	text	text	varchar(240)		
projectName	text	text	varchar(240)		
percentage	text	text	varchar(240)		
userId	text	text	varchar(240)		
date	text	text	varchar(240)		
totalCorrect	int	int	number		
numberPhases	int	int	number		

The thread table provides the all the information of the file. Also provides the percentage and total correct which are calculated by ReMoTe. Besides, this table records the date of submission.

Table 4. Setting Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	

companyName	text	text	varchar(240)		
projectName	text	text	varchar(240)		
bugzilla	text	text	varchar(240)		Location of bugzilla
message	text	text	varchar(240)		Message to all the team numbers
numberIterations	int	int	number		

The setting table provides the information of project manager has to set up, such as the number of iterations, message to the whole team and the location of Bugzilla which is an URL.

Table 5. Set View Projects Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
location	text	text	varchar(240)		Server location
companyName	text	text	varchar(240)		
projectName	text	text	varchar(240)		
dbType	int	int	number		Database type
username	text	text	varchar(240)		Database username

password	text	text	varchar(240)		
personId	int	int	number		Id number of the view person
databaseName	text	text	varchar(240)		

The setViewProjects table provides the detailed information of the project that are the management wants to view. They include the server's location, company name, project name, database type, database name, user name, view person Id and password.

Table 6. Set Team Names Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
companyName	text	text	varchar(240)		
projectName	text	text	varchar(240)		
iteration	text	text	varchar(240)		
teamId	int	int	number		
teamName	int	int	number		

The setTeamNames table provides the team name and other necessary information.

Table 7. Set Number Phases Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
numberPhases	int	int	number		
userId	int	int	number		
iteration	int	int	number		

The setNumberPhases table provides the number phases and other necessary information.

Table 8. Set Dates Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
phaseNumber	int	int	number		
companyNmae	text	text	varchar(240)		
projectName	text	text	varchar(240)		

phaseName	text	text	varchar(240)		
startYear	int	int	number		
startMonth	int	int	number		
startDay	int	int	number		
phaseDays	text	text	varchar(240)		
iteration	int	int	number		
userId	int	int	number		

The setDates table provides the detail data for the estimated delivery day such as project start year, start month, start day and estimated accomplish period.

Table 9. Select Team Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
managerId	int	int	number		
subUserId	int	int	number		
projectName	text	text	varchar(240)		
iteration	int	int	number		
teamName	int	int	number		

The selectTeam table provides information of the user who has been selected by project manager or team manager.

Table 10. Message Board Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
ID	int	int	number	PRI	
userId	int	int	number		
projectName	text	text	varchar(240)		
date	text	text	varchar(240)		
broadcast	int	int	number		
replyId	int	int	number		The id number of the reply message for the original message
subject	text	text	varchar(240)		
message	text	text	varchar(240)		

The MessageBoard table provides the information of the message board such as date, broadcast, replay ID, subject and the content of the message.

Table 11. Uniform Resource Locator Table

Field Name	Type			Key	Extra
	mySQL	Access	Oracle		
URL	text	text	varchar(240)		

The URL table provides the URL information which is what URL has been using currently.

3.3 Controller Design

Figures 26 to 28 show the class diagram for the controller classes. There are nine classes, which are AdminController class's child class, twenty-one classes inherit from InterfaceController class and four classes inherit from ThreadController class.

Child classes of AdminController. See Figure 26:

- AcceptUserAction connects the database and updates the information to the user table and generates an email to the user's email box.
- DeleteMessageAction connects the database and though the function in MessageBoardDAO to clear the message in the messageBoard table.

- DeleteUserAction connects the database and though the function in the UserDAO to clear the user information in the user table.
- EditMessagesAction returns to the editMessages HTML page to let the user to make changes.
- EditUserAction connects to the database find the user's id then returns to the editRegisteration HTML page for that user.
- EditUserPageAction connects to the database and let the user make changes about his/her information. If there is no information in the database then it will return to edigRegisteration HTML page.
- LoginAction connects to the configuration file, confirm the user name and password then verify the login. Returns to the login page if they are not match.
- RegisterationAction connects to the database and keeps all the information of users in the user table.
- SetViewOnlyAction connects the database and gets that user by the user Id, then sets the user type from N to V in the user table and generates an email to that user's email box.

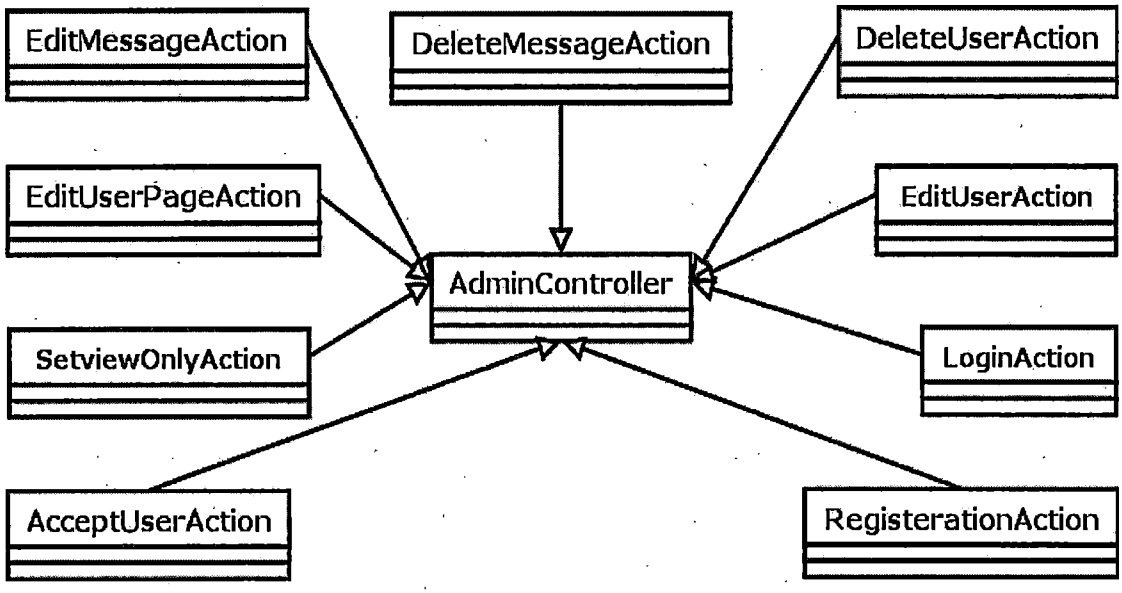

Figure 26. Class Diagram for AdminController

Child classes of InterfaceController. See Figure 26:

- AddThreadAction connects to the database and adds the thread to the thread table.
- CreateThreadAction is the error-protected class. It will return to the createThread HTML file.
- DeleteFileAction connects to the database to delete the file info from the thread table and stats table.
- EditUserAction connects to the database and let the user to make changes about his/her information. If there is no information in the database then it will return to

editRegisteration HTML page. Also it will generate error message if it can not connect to the database.

- ForgotPasswordAction connects to the database and finds the user by the user's email, and then send an email to inform the user about the password. Also, it will show the error message on the interface if the user name or the email is incorrect.
- GanttChartAction returns to the ganttChart HTML file.
- LoginAction connects to the database and verifies the user name and password, if they are verified then returns to the intro page, otherwise gives the error message and returns back to the login page.
- MessageBoardAction connects to the database and updates the messageBoard table.
- RegistrationAction connects to the database and updates the user table.
- SelectTeamAction connects to the database and updates the selectTeam table.
- SetDatesAction connects to the database and updates the setDates table.
- SetNumberPhasesAction connects to the database and updates the setNumberPhases table.

- SetPictureAction connects to the database and updates the user table.
- SetPrototypeAction uses session to set to prototype and returns to intro page.
- SetTeamNamesAction connects to the database and updates the setTeamNames table.
- SettingsAction connects to the database and updates the settings table.
- SetViewProjectsAction connects to the database and updates the setViewProjects table.
- StatisticsAction returns to statistics HTML file.
- ViewMessageAction returns to the viewMessage HTML file.
- ViewMessagesAction returns to the viewMessages HTML file.

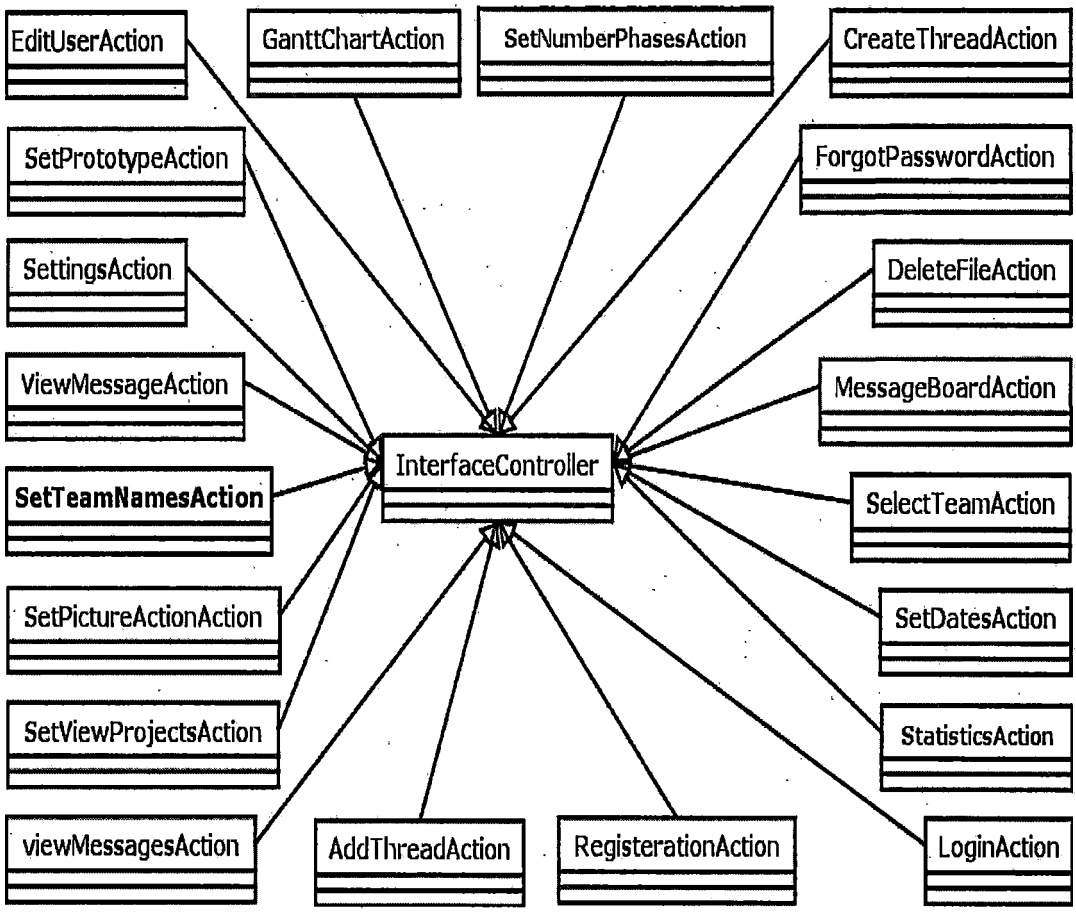

Figure 27. Class Diagram for InterfaceController

Child classes of ThreadController. See Figure 27:

- GanttChartAction returns to the ganttChart HTML file.
- ShowFileAction returns to the showFile HTML file.
- ShowNotesAction connects to the database and updates the fileNotes table.
- StatisticsAction returns to the statistic HTML file.

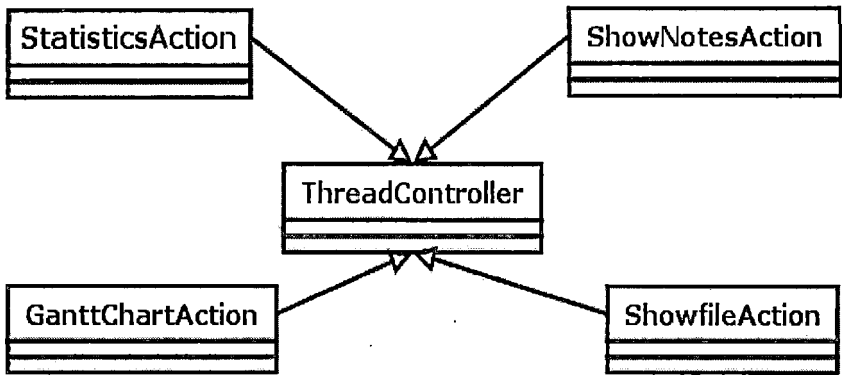

Figure 28. Class Diagram for ThreadController

3.4 View Design

Shown as Figure 29, View is separated to admin, client, and thread portion. They respectively have eight, nineteen, and nine files.

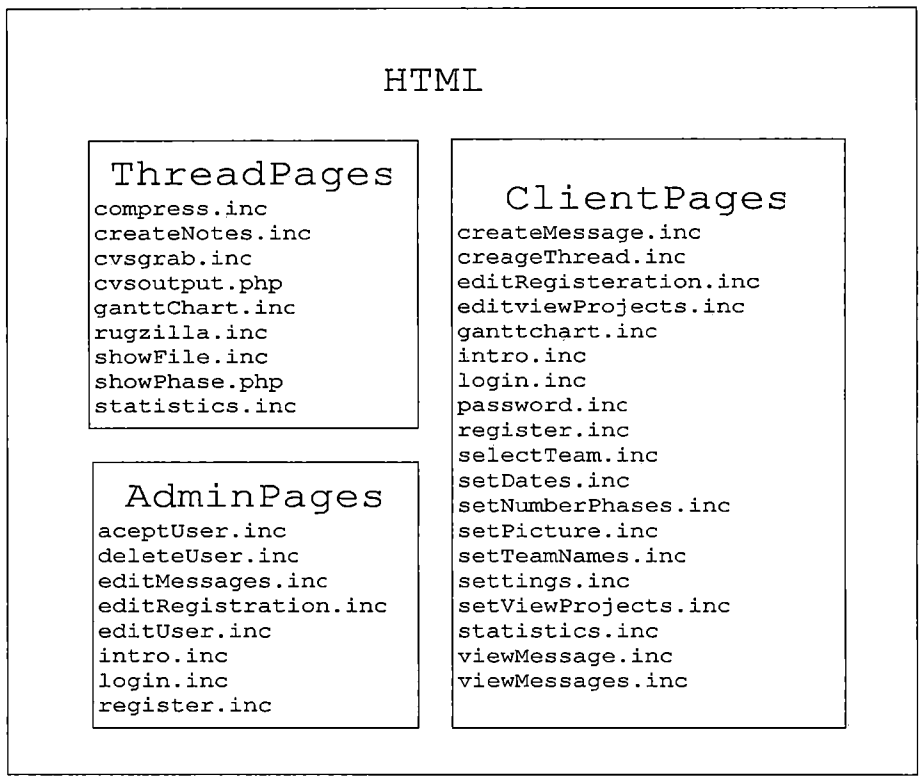

Figure 29. Graphical User Interface Architecture

3.5 Detailed Design

In this project used ODBC Socket Server for connecting Access database. After installing the ODBC Socket Server, setup the Data Sources (ODBC) in Windows servers, then add Microsoft Access Driver. For making the connection works like mySQL and other databases, an ODBC connection class named ODBCSocketServer is created. This class includes most functions in the PHP library for mySQL database such as

connect, close, select database, query, fetch row, seek row, error. Also included are open port, parse XML etc.

In the model design component, ConnectDAO is the key class for connecting three databases with all other DAO classes, shown as Figure 30. Instead of connecting to database directly from these DAO files, connect to the ConnectDAO file first, afterwards it diverges to different databases connection in ConnectDAO file. This class made ReMoTe linked to three databases possible without changing exist functions. Also in this class, because of the syntaxes of each database is all slightly different, a set of regular expression function has been used to replace the different SQL.

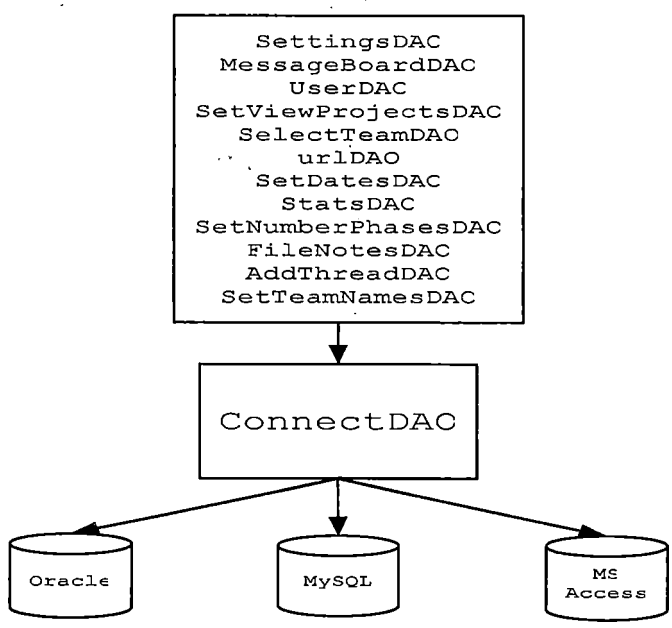

Figure 30. ConnectDAO Class

In the controller design component, there is one class called SetViewOnlyAction that has been created in admin for letting the administrator setup the viewer's information. There is also another class called SetViewProjectsAction that has been created in client for letting the user to setup the information for which project to view. Both of them are described in section 3.3.

In the view design component, there are two pages that are created which are setViewProjects and editViewProjects. The former is a GUI for the user to set up the project information; the latter is a GUI for the user to modify the project's information.

Displaying all the information of the databases from different location can be viewed in a single page at the same time, URL class has been conducted. This class would make an URL table to keep the local URL location.

For MS Access database, it is different with other database which has a functionality to sort all the items of the table in alphabet order no matter what order had been put in SQL. To avoid this confusion and match up with other databases, in SQL statement, alphabets have been put in front of each column name in order for Access.

Also in Oracle there are several reserved word such as user and date. For avoiding the confusion, they had to be changed to other names.

CHAPTER FOUR

SOFTWARE QUALITY ASSURANCE

4.1 Unit Test Plan

4.1.1 Test 1 - Javascript

Purpose of this test is verifying the following on each web page:

1. Links
    - All necessary links exist on page
    - All links work correctly
2. Web Pages
    - Verify all web pages are used
    - Verify no additional web pages are necessary

Procedure of this test is following:

- Open Internet Explorer or another browser
- Have a hardcopy of each web page nearby for mark-up if necessary
- Open each web page in the list above and perform the following checks
    a. Insure the banner is correct for the page
    b. Insure only the appropriate graphics are present (menus, buttons, etc.)

c. Check the descriptions on the graphics for brevity and correctness

d. Check spelling

e. Insure all terminology will be easily understood by the common user

f. Verify all necessary links to other pages exist

g. Check each link for operation

- Repeat Step c for all web pages.

Each web page passed the test.

4.1.2 Test 2 - Program

Purpose of this test is following:

1) This test will verify the functionality of the following sections of code:

a. html code section

b. src code section

c. static code section

2) The following interfaces will be tested:

a. Code section with database

b. Code section with user interface

c. Code section with other code sections

Procedure of this test is following:

1. Refer to the ReMoTe: Software Requirements Specification External Interface section to identify the interfaces required (database access, user entry, or function call).
2. Execute each section of compiled code using breakpoints and single step execution when necessary.
3. Verify interface was successful at each step.
4. If possible, correct any code issues and retest to verify results.
5. If immediate correction and retest is not practical, make detailed notes of problem and fix on hardcopy of code section.
6. Repeat Step b for all code sections.

Single stepping through the code will allow all conditional, branches, and function calls to be verified.

4.1.3 Test 3 - Database

Purpose of this test is to test if data have been contained and what data have been contained in each following tables:

- User
- Settings
- SetDates
- SetTeamNames
- SetNumberPhases

- SelectTeam
- MessageBoard
- SetViewProjects
- Threads
- Statistics
- URL

Procedure of this test is following:

- Input command to show tables
- Input description command to show in each table what is the data filed and data type
- Input select command to show the data in each table

Expected result is showing on table 12.

Table 12. Result of Database Unit Test

Forms	Results
User table	Passed
Settings table	Passed
SetDates	Passed
SetTeamNames	Passed
SetNumberPhases	Passed
SelectTeam	Passed
Messageboard	Passed

SetViewProjects	Passed
Thread	Passed
Statistics	Passed
URL	Passed

Shown as table 12, every table passed the test.

4.2 Integration Test Plan

After the modules have been fully tested components are ready to be combined. The integration test strategies for ReMoTe are regarded as thread testing.

The purpose of the Integrated Test is to test the full functionality of the ReMoTe system. There are four types of users:

1) Regular User
2) System Administrator
3) Project Manager
4) Management

There are several functions available to users based on login privileges. The Integrated Test will test the available functions for each login, and therefore will be

subdivided into four tests, one for each login. Expected result is all the functions passed the test.

4.3 System Test Plan

The system test tests for the following criteria as needed.

- Hyperlink

The hyperlinks are checked many times to confirm that the links are working consistently.

- Content

The Web pages are entered in and out many times. Each time the Web page is consistent.

- Consistent Look and Feel

The Web pages are entered in and out many times. Each time the Web pages are consistent and easy to navigate.

- Performance

The Web pages are entered in and out many times. Each time the Web pages download within five to ten seconds.

- Submit

Users are able to save the data and have it entered into the database. This had been done a number of times. For the user entered the wrong data, ReMoTe has the warning or error message for each wrong input. Also, ReMoTe is able

to continue without crashing even though the user entered a site that does not exist. Figures 31 to 34 show the protection warning pages.

Additionally, mySQL database part was tested by Algorithma Project which is the project in the CSCI 455 (Software Engineering) course in winter 2005.

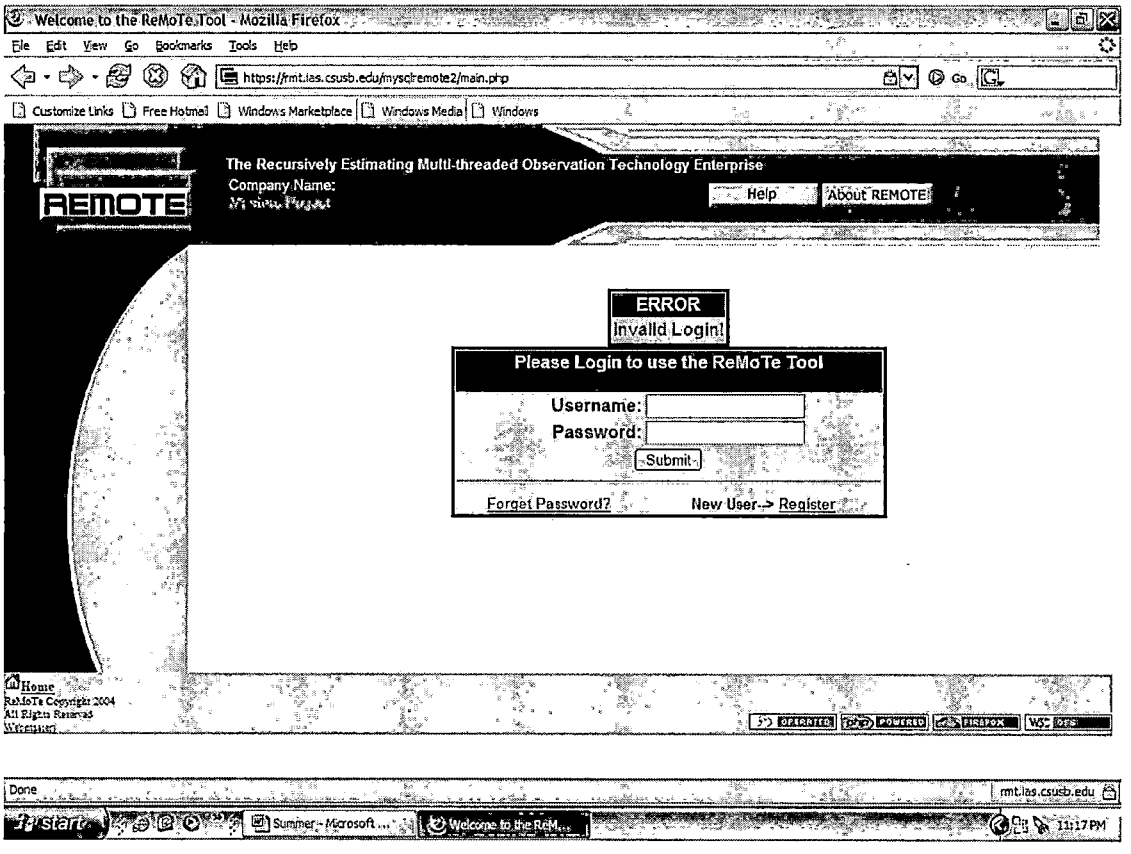

Figure 31. Invalid Login Page

This page protects the server from crashing when the user enters a wrong user name or password.

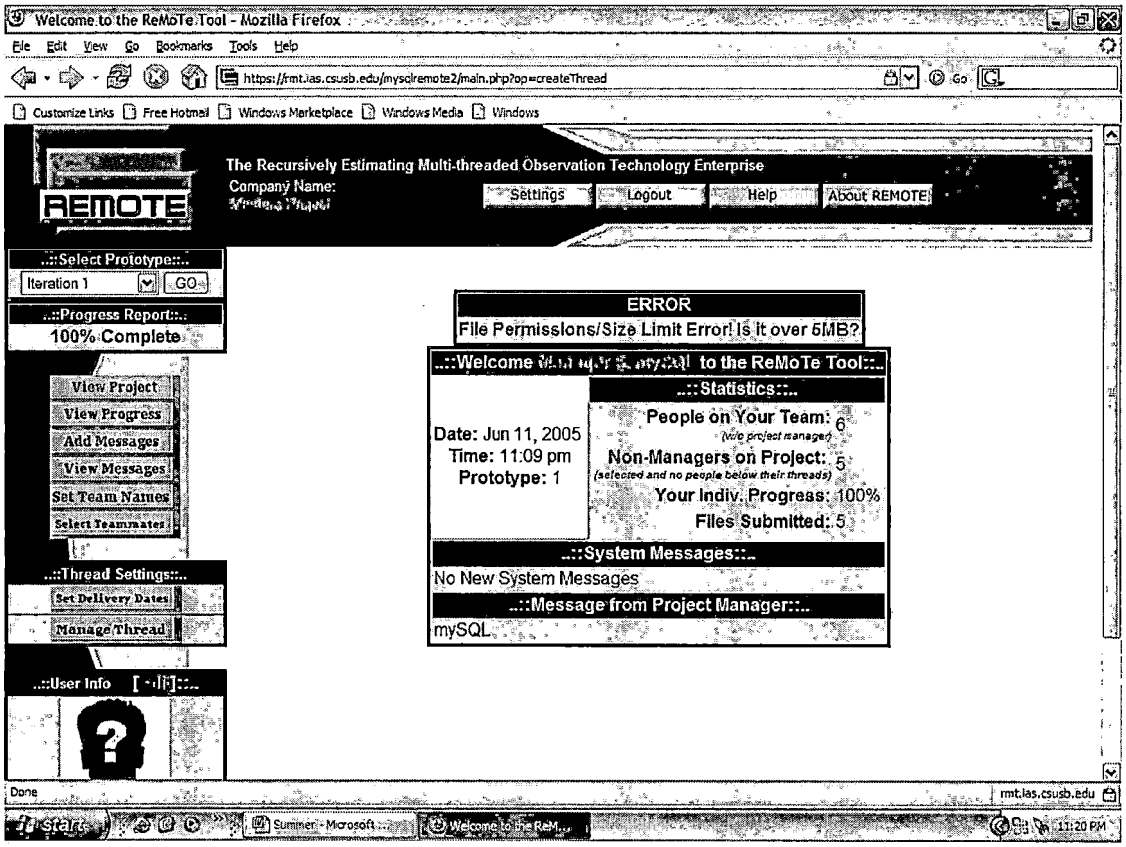

Figure 32. Invalid Thread Page

This page is to limit the size of the threads that the user can submit. It also protects the server from crashing when the user enters an invalid file.

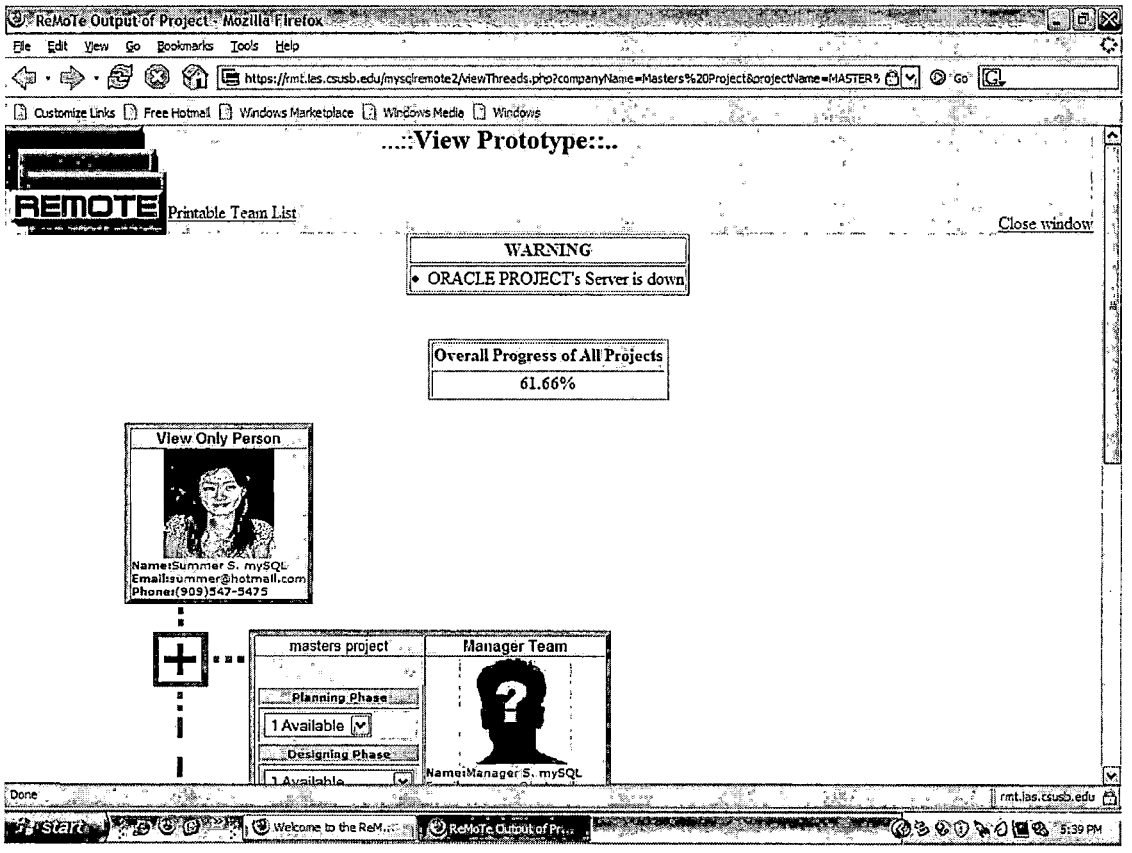

Figure 33. Invalid Server Page

This page protects the server from crashing when the server is down.

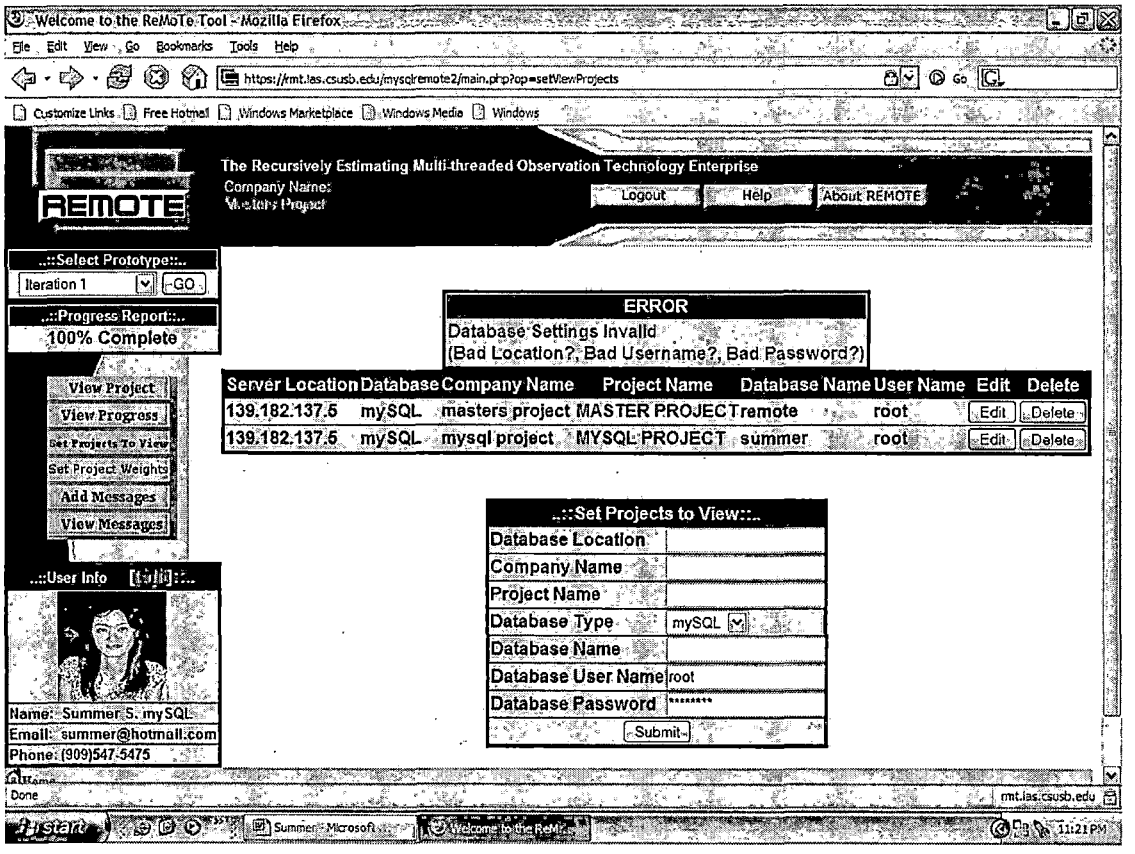

Figure 34. Invalid Database Info Page

This page protects the server from crashing when the user enters invalid database information, such as wrong database name, password or server location.

CHAPTER FIVE

MAINTENANCE

5.1 Software Installation Process

In this project, it requires Gentoo Linux, Windows XP, Oracle database, Oracle Client, Apache, PHP, mySQL database, ODBC Socket Server to execute the program. There are four steps in the installation process.

Step 1

- Install Gentoo Linux, Apache, PHP and mySQL database, oracle client.
- Install Windows XP, Oracle database.
- Set up Oracle database:

  a. Go to:

     http://otn.oracle.com/software/tech/oci/instantclient/htdocs/linuxsoft.html

     Download the Basic client package with SDK, which are:

     oracle-instantclient-basic-10.1.0.3-1.i386.rpm

     oracle-instantclient-devel-10.1.0.3-1.i386.rpm

  b. Put them in /usr/portage/distfiles

  c. In the terminal run:

     ACCEPT_KEYWORDS="~x86" emerge=dev-db/oracle-instantclient-basic-10.1.0.3

d. Run: USE="gd odbc oracle oci8" emerge php mod_php

- Install ODBC Socket Server.
- Set up ODBC Socket Server:

a. Run the following command via the run option on the Windows NT start menu to setup <path to ODBCSocketServer.exe>\ODBCSocketServer.exe /Service

b. Enter information about ODBC Socket Server into the database for configuration.

Step 2

Store all HTML and JavaScript files.

Step 3

Create database instances in each database.

Step 4

Store all the PHP files under the directory that was defined in the path of the Web server.

5.2 Source Code

In this version of ReMoTe, there are six PHP classes, two PHP files and two HTML pages which are created for multi-database function. All the source codes and structures are list on the Appendix A. Also they are copied in attached CD.

The following is the list of the source files for multi-database.

- editViewProjects.inc
- setViewProjects.inc
- SetViewOnlyAction.php
- SetViewProjectsAction.php
- ConnectDAO.php
- ODBCSocketServer.php
- SetViewProjects.php
- SetViewProjectsDAO.php
- URL.php
- urlDAO.php

5.3 Improvement

For the PHP 5.0 version, there is a function called dbx_connect, using this function can connect to mySQL, ODBC, PostgreSQL, MSSQL, FrontBase, Sybase, Oracle and SQLite without making any extra functions. The future implementer can test this function to make ReMoTe more simple, flexible and powerful.

CHAPTER SIX

CONCLUSIONS AND FUTURE DIRECTIONS

6.1 Conclusions

For software development companies using ReMoTe Tool with a need to access multiple sub-projects that are being developed in geographically different located sites, this project is the solution. A project manager can remotely access project updates irregardless of database type or location.

ReMoTe tool is a Web-based application. It is capable of inserting, deleting, updating, selecting and displaying all Id event information on a software project anywhere. The user can immediately see the effects of changes at a project level. Or, the user can drill down for details on any team's task level. ReMoTe can calculate both individuals and teams percentage of progress. Having CVS link, support for team work or management of long term development of a huge and complex software project is very easy. Also, ReMoTe maintains the historical logging of all software artifacts. Additionally, the overall project progress is calculated and displayed graphically. By changing the configuration, users can set up their company name, database type, database location, user name and password. It also provides an

interface for users who intend to view the overall progress without difficulty or database limitation. ReMoTe keeps the project under control at all times.

Comparing ReMoTe with Microsoft Project, ReMoTe is more detailed on personal progress management that is built up by each member of the project. MS Project is more focused on the entire and independent management without coordination with other members. MS Project starts from defining the project, building the plan and adjusting the plan. Also included is the estimate working time, control budget, and communication. In MS Project, it would calculate task start and finish dates and durations automatically, but in ReMoTe, they are all estimated by the users themselves. Table 13 compared the different features between ReMoTe and MS Project.

Table 13. ReMoTe and Microsoft Project Comparison

Features	ReMoTe	MS Project
1. Organizing and retrieval of software artifacts	yes	no
2. Estimating overall progress of software project	yes	yes

3. Displaying individual progress	yes	no
4. Scheduling events	no	yes
5. critical path analysis	no	yes
6. Supporting Multi-Database	yes	no
7. Transparent and real time accomplishment report	yes	no
8. Group participated and well communicated functionality	yes	no
9. Customize and adjust plans	no	yes

ReMoTe is a very unique software process management tool compared with other management tools. For anyone who considers Microsoft Project is taking complete care of everything, then ReMoTe is a good choice for them. For using ReMoTe, there are multi-database choices. Access is part of Microsoft Office. It is easily available for most of the people using windows, the learning curve for Access is not as challenging as it is for other database SQL. MySQL is an open source database, easier for anyone to use and Oracle is an enterprise database which fit large companies. Additionally, ReMoTe has been tested by students of the CSCI 455 (Software Engineering) course for two quarter terms. During the evaluation period time, ReMoTe ran considerably

stable and made the project management of the CSCI455 class easier to manage all the teams and each individual's achievement and progress.

6.2 Future Directions

ReMoTe was designed to support any kind of database, but in this project, there are three kinds of database supported, i.e., MySQL, Oracle, and MS Access. Since having the multi-database built, it is relatively easy to add other kind of database systems such as FrontBase, PostgreSQL, OpenBase, Sybase, DB2 etc.

Also, there is another function that can be added, which is the management of multiple projects. Using this function, ReMoTe can find the latest time (completion time) for the entire project. The multiple projects have precedence relation, that is, an individual project can not be started until all projects that preceded it are all finished. Figure 35 shows the estimated delivery time in a huge project. Using the critical path analysis algorithm, the delivery of 44 days and the path from Project 1 to Project 3 to Project 6 to Project 7 are computed.

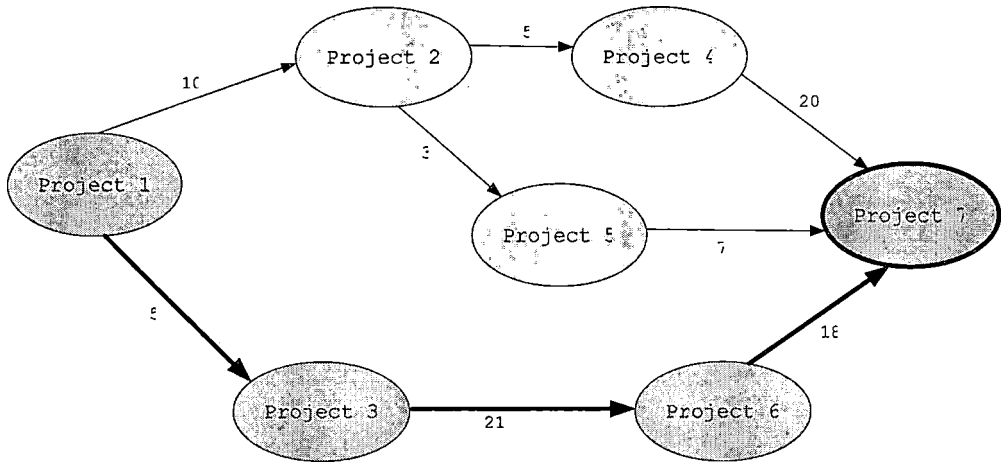

Figure 35. Estimated Delivery Route 1

Figure 36 shows when the estimate delivery times were changed, ReMoTe could automatically recompute the delivery path. In this example, the delivery time is 45 days and the path is from Project 1 to Project 2 to Project 5 to Project 7.

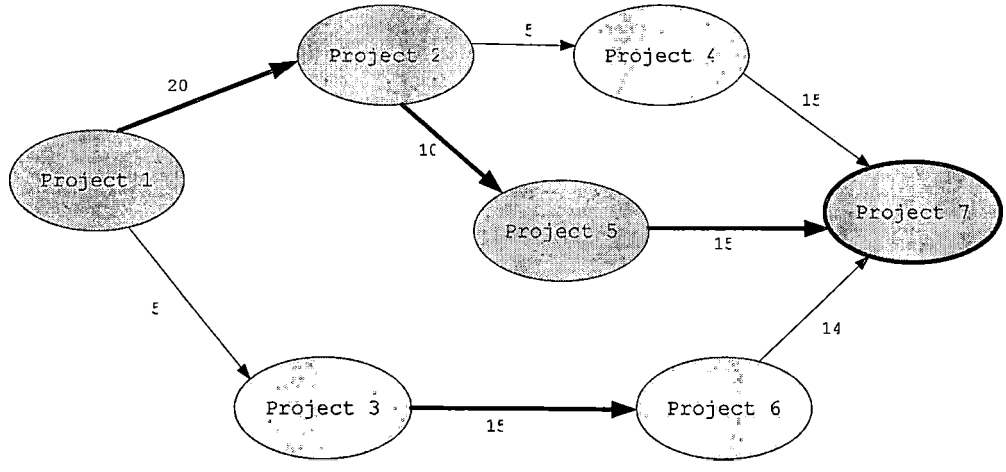

Figure 36. Estimated Delivery Route 2

APPENDIX A

LIST OF SOURCE CODE FILES

```
ReMoTe
|-- GanttChart.php
|-- WEB-INF
|   |-- html
|   |   |-- adminPages
|   |   |   |-- acceptUser.inc
|   |   |   |-- deleteUser.inc
|   |   |   |-- editMessages.inc
|   |   |   |-- editRegistration.inc
|   |   |   |-- editUser.inc
|   |   |   |-- intro.inc
|   |   |   |-- layout
|   |   |   |   |-- admin.tpl
|   |   |   |   `-- logout.tpl
|   |   |   |-- login.inc
|   |   |   `-- register.inc
|   |   |-- clientPages
|   |   |   |-- createMessage.inc
|   |   |   |-- createThread.inc
|   |   |   |-- editRegistration.inc
|   |   |   |-- editViewProjects.inc
|   |   |   |-- ganttChart.inc
|   |   |   |-- intro.inc
|   |   |   |-- layout
|   |   |   |   |-- client.tpl
|   |   |   |   `-- logout.tpl
|   |   |   |-- login.inc
|   |   |   |-- password.inc
|   |   |   |-- register.inc
|   |   |   |-- selectTeam.inc
|   |   |   |-- setDates.inc
|   |   |   |-- setNumberPhases.inc
|   |   |   |-- setPicture.inc
|   |   |   |-- setTeamNames.inc
|   |   |   |-- setViewProjects.inc
|   |   |   |-- settings.inc
|   |   |   |-- statistics.inc
|   |   |   |-- viewMessage.inc
|   |   |   `-- viewMessages.inc
|   |   `-- threadPages
|   |   |   |-- compress.inc
|   |   |   |-- createNotes.inc
|   |   |   |-- cvsgrab.inc
|   |   |   |-- cvsoutput.php
|   |   |   |-- ganttChart.inc
|   |   |   |-- rugzilla.php
|   |   |   |-- showFile.inc
```

```
|   |       |-- showFile.php
|   |       |-- showPhase.php
|   |       |-- showTeamList.inc
|   |       |-- statistics.inc
|   |       |-- threadLayout
|   |       |   |-- bugzilla.tpl
|   |       |   `-- thread.tpl
|   |       `-- thread_top.php
|   |-- src
|   |   |-- admin
|   |   |   |-- AcceptUserAction.php
|   |   |   |-- DeleteMessageAction.php
|   |   |   |-- DeleteUserAction.php
|   |   |   |-- EditMessagesAction.php
|   |   |   |-- EditUserAction.php
|   |   |   |-- EditUserPageAction.php
|   |   |   |-- LoginAction.php
|   |   |   |-- RegistrationAction.php
|   |   |   `-- SetViewOnlyAction.php
|   |   |-- client
|   |   |   |-- AddThreadAction.php
|   |   |   |-- CreateThreadAction.php
|   |   |   |-- DeleteFileAction.php
|   |   |   |-- EditUserAction.php
|   |   |   |-- ForgotPasswordAction.php
|   |   |   |-- GanttChartAction.php
|   |   |   |-- LoginAction.php
|   |   |   |-- MessageBoardAction.php
|   |   |   |-- OverridePhaseNamesAction.php
|   |   |   |-- RegistrationAction.php
|   |   |   |-- SelectTeamAction.php
|   |   |   |-- SetDatesAction.php
|   |   |   |-- SetNumberPhasesAction.php
|   |   |   |-- SetPictureAction.php
|   |   |   |-- SetPrototypeAction.php
|   |   |   |-- SetTeamNamesAction.php
|   |   |   |-- SetViewProjectsAction.php
|   |   |   |-- SettingsAction.php
|   |   |   |-- StatisticsAction.php
|   |   |   |-- ViewMessageAction.php
|   |   |   `-- ViewMessagesAction.php
|   |   |-- controller
|   |   |   |-- AdminController.php
|   |   |   |-- InterfaceController.php
|   |   |   `-- ThreadController.php
|   |   |-- database
|   |   |   |-- AddThread.php
```

```
|   |   |   |-- AddThreadDAO.php
|   |   |   |-- ConnectDAO.php
|   |   |   |-- Database.php
|   |   |   |-- FileNotes.php
|   |   |   |-- FileNotesDAO.php
|   |   |   |-- MessageBoard.php
|   |   |   |-- MessageBoardDAO.php
|   |   |   |-- ODBCSocketServer.php
|   |   |   |-- SelectTeam.php
|   |   |   |-- SelectTeamDAO.php
|   |   |   |-- SetDates.php
|   |   |   |-- SetDatesDAO.php
|   |   |   |-- SetNumberPhases.php
|   |   |   |-- SetNumberPhasesDAO.php
|   |   |   |-- SetTeamNames.php
|   |   |   |-- SetTeamNamesDAO.php
|   |   |   |-- SetViewProjects.php
|   |   |   |-- SetViewProjectsDAO.php
|   |   |   |-- Settings.php
|   |   |   |-- SettingsDAO.php
|   |   |   |-- Stats.php
|   |   |   |-- StatsDAO.php
|   |   |   |-- ThreadDAO.php
|   |   |   |-- URL.php
|   |   |   |-- User.php
|   |   |   |-- UserDAO.php
|   |   |   `-- urlDAO.php
|   |   `-- thread
|   |       |-- GanttChartAction.php
|   |       |-- ShowFileAction.php
|   |       |-- ShowNotesAction.php
|   |       |-- ShowTeamListAction.php
|   |       `-- StatisticsAction.php
|   `-- static
|       |-- files
|       |-- help
|       |   |-- About.php
|       |   |-- README.html
|       |   |-- frameMain.html
|       |   |-- helpFiles
|       |   |   |-- FAQ.html
|       |   |   |-- createcvs.html
|       |   |   |-- cvstutorial.html
|       |   |   |-- deadlines.html
|       |   |   |-- images
|       |   |   |   |-- accept.jpg
|       |   |   |   |-- addmessage.jpg
```

```
|   |   |   |   |-- addthread.jpg
|   |   |   |   |-- addthread1.jpg
|   |   |   |   |-- adminlogin.jpg
|   |   |   |   |-- adminside.jpg
|   |   |   |   |-- change.jpg
|   |   |   |   |-- colorsheme.jpg
|   |   |   |   |-- contact.jpg
|   |   |   |   |-- databaseid.jpg
|   |   |   |   |-- deadlines.jpg
|   |   |   |   |-- editthread.jpg
|   |   |   |   |-- engineers.jpg
|   |   |   |   |-- forgot.jpg
|   |   |   |   |-- intro.jpg
|   |   |   |   |-- login.jpg
|   |   |   |   |-- logout.jpg
|   |   |   |   |-- mainmanager.jpg
|   |   |   |   |-- messageboard.jpg
|   |   |   |   |-- messageboard1.jpg
|   |   |   |   |-- register.jpg
|   |   |   |   |-- register2.jpg
|   |   |   |   |-- selectTeam.jpg
|   |   |   |   |-- setdates.jpg
|   |   |   |   |-- setdates1.jpg
|   |   |   |   |-- setdates2.jpg
|   |   |   |   |-- settings.jpg
|   |   |   |   |-- stats.jpg
|   |   |   |   |-- stats.php.jpeg
|   |   |   |   |-- stats2.php.jpeg
|   |   |   |   |-- subnames.jpg
|   |   |   |   |-- subnames1.jpg
|   |   |   |   |-- teammanagers.jpg
|   |   |   |   |-- testing.jpg
|   |   |   |   |-- viewproto.jpg
|   |   |   |   |-- waterfall.jpg
|   |   |   |   |-- weights.jpg
|   |   |   |   |-- weights1.jpg
|   |   |   |   `-- weights3.jpg
|   |   |   |-- install.html
|   |   |   |-- introduction.html
|   |   |   |-- messageboard.html
|   |   |   |-- php.ini
|   |   |   |-- projectname.html
|   |   |   |-- ref.inc
|   |   |   |-- registration.html
|   |   |   |-- remoteadmin.html
|   |   |   |-- rugzilla.html
|   |   |   |-- selectteam.html
```

```
|       |   |   |-- setup.html
|       |   |   |-- side.html
|       |   |   |-- submitthreads.html
|       |   |   |-- teamnames.html
|       |   |   |-- top.html
|       |   |   `-- viewthreads.html
|       |   `-- main.html
|       |-- images
|       |   |-- DeMeloDarrion.jpg
|       |   |-- DoeJohn.jpg
|       |   |-- DrConcepcion.jpg
|       |   |-- about.gif
|       |   |-- acceptUser.swf
|       |   |-- addBug.gif
|       |   |-- addThread.gif
|       |   |-- addbug.gif
|       |   |-- base.gif
|       |   |-- bugzilla.jpg
|       |   |-- cd.gif
|       |   |-- color-scheme.jpeg
|       |   |-- createMessage.jpg
|       |   |-- csci.gif
|       |   |-- css.png
|       |   |-- cube_blue.gif
|       |   |-- cube_green.gif
|       |   |-- cube_red.gif
|       |   |-- delUsers.gif
|       |   |-- deleteUser.swf
|       |   |-- deliveryDates.jpg
|       |   |-- delmessages.gif
|       |   |-- edit.gif
|       |   |-- editMessage.swf
|       |   |-- editUser.swf
|       |   |-- empty.gif
|       |   |-- favicon.ico
|       |   |-- firefox.png
|       |   |-- folder.gif
|       |   |-- folderopen.gif
|       |   |-- frown.gif
|       |   |-- gantt.gif
|       |   |-- ganttChart.gif
|       |   |-- gentoo.png
|       |   |-- globe.gif
|       |   |-- grabber.gif
|       |   |-- grin.gif
|       |   |-- help.gif
|       |   |-- home.gif
```

```
|         |    |-- imgfolder.gif
|         |    |-- index.html
|         |    |-- join.gif
|         |    |-- joinbottom.gif
|         |    |-- joriz.jpg
|         |    |-- line.gif
|         |    |-- linemore.gif
|         |    |-- lines.gif
|         |    |-- logout.gif
|         |    |-- manageThread.jpg
|         |    |-- messages.jpg
|         |    |-- minus.gif
|         |    |-- minusbottom.gif
|         |    |-- musicfolder.gif
|         |    |-- nolines_minus.gif
|         |    |-- nolines_plus.gif
|         |    |-- notSet.jpg
|         |    |-- page.gif
|         |    |-- percent.jpg
|         |    |-- php.png
|         |    |-- plus.gif
|         |    |-- plusbottom.gif
|         |    |-- progress.jpg
|         |    |-- projectManager.swf
|         |    |-- question.gif
|         |    |-- remote.ico
|         |    |-- remotelogo.gif
|         |    |-- rugzilla.gif
|         |    |-- selThreads.jpg
|         |    |-- setProjects.jpg
|         |    |-- settings.gif
|         |    |-- sidebar.jpg
|         |    |-- small-rmt.gif
|         |    |-- smile.gif
|         |    |-- summer.jpg
|         |    |-- teamNames.jpg
|         |    |-- teamweights.gif
|         |    |-- top_banner.jpg
|         |    |-- trash.gif
|         |    |-- viewProject.jpg
|         |    `-- warning.gif
|         `-- scripts
|              |-- dtree.css
|              |-- dtree.js
|              `-- globals.css
|-- adminPage.php
|-- index.html
```

```
|-- main.php
|-- settings
|   `-- setup.conf
|-- statistics.php
`-- viewThreads.php
```

APPENDIX B

JAVASCRIPT SOURCE CODE FILES

```
<?php
$setViewProjectsDAO = new SetViewProjectsDAO;
$viewShow = $setViewProjectsDAO->get_viewProjects($userShow);

 print "<table rules='rows' class='show' rules='rows'>
          <tr>
            <th>
                <b>Server Location</b>
            </th>
            <th>
                <b>Database</b>
            </th>
            <th>
                <b>Company Name</b>
            </th>
            <th>
                <b>Project Name</b>
            </th>
            <th>
                <b>Database Name</b>
            </th>
            <th>
                <b>User Name</b>
            </th>
                <th>
                <b>Edit</b>
            </th>
                <th>
                <b>Delete</b>
            </th>
          </tr>
  ";
      if($viewShow != null){
      for($i = 0; $i < sizeOf($viewShow); $i++){
      if($viewShow[$i]->get_dbType() == "1"){
            $dbType = "mySQL";
      }else if($viewShow[$i]->get_dbType() == "2"){
       $dbType = "Access";
      }else if($viewShow[$i]->get_dbType() == "3"){
       $dbType = "Oracle";
      }
      print "<tr>
            <td>
                ".$viewShow[$i]->get_location()."
            </td>
            <td>
                ".$dbType."
            </td>
            <td>
                ".$viewShow[$i]->get_companyName()."
            </td>
            <td>
                ".$viewShow[$i]->get_projectName()."
            </td>
            <td>
                ".$viewShow[$i]->get_databaseName()."
            </td>
```

```
            <td>
                ".$viewShow[$i]->get_userName()."
            </td>
            <td>
            <form name = 'edit$i' action='$PHP_SELF' method='post'>
             <input type='hidden' name='op' value='editViewProjects'>
             <input type='hidden' name='type' value='edit'>
             <input type='hidden' name='viewProjectId'
value='".$viewShow[$i]->get_Id()."'>
             <input type='submit' value='Edit'>
            </form>
            </td>
            <td>
            <form name = 'delete$i' action='$PHP_SELF' method='post'>
                        <input type='hidden' name='op'
value='deleteViewProjects'>
                        <input type='hidden' name='type' value='delete'>
                        <input type='hidden' name='viewProjectId'
value='".$viewShow[$i]->get_Id()."'>
                        <input type='submit' value='Delete'>
                    </form>
            </td>
          </tr>";
      }
      }else print "<tr><td colspan=7>No Projects Set By
Person</td></tr>";
      print "</table>";

?>
<br><br>
<table class='show' rules='rows'>
      <tr>
            <th bgcolor="#000000" colspan=4>
                  <form name='RMT' method='post' action='<?print
$PHP_SELF?>'>
                  <font color=white><b>..::Set Projects to
View::..</b></font>
            </th>
      </tr>
        <tr>
                  <td>
                  <b>Database Location</b>
            </td>
                  <td>
                  <input type="text" name="location">
            </td>
      </tr>
        <tr>
                  <td>
                  <b>Company Name</b>
            </td>
                  <td>
                  <input type="text" name="companyName">
            </td>
      </tr>
        <tr>
                  <td>
```

```
                    <b>Project Name</b>
              </td>
                    <td>
                    <input type="text" name="projectName">
              </td>
        </tr>
          <tr>
                    <td>
                    <b>Database Type</b>
              </td>
                    <td>
                    <select name="dbType">
                    <option value="1">mySQL</option>
                    <option value="2">Access</option>
                    <option value="3">Oracle</option>
                    </select>
              </td>
        </tr>
          <tr>
                    <td>
                    <b>Database Name</b>
              </td>
                    <td>
                    <input type="text" name="databaseName">
              </td>
        </tr>
          <tr>
                    <td>
                    <b>Database User Name</b>
              </td>
                    <td>
                    <input type="text" name="userName">
              </td>
        </tr>
          <tr>
                    <td>
                    <b>Database Password</b>
              </td>
                    <td>
                    <input type="password" name="password">
              </td>
        </tr>
        <tr>
                <td colspan=2 align='center' >
                    <input type="hidden" name="op"
value="submitViewProjects">
                    <input type="hidden" name="userId" value="<?print
$userShow->get_userId()?>">
                    <input type="submit" value="Submit">

              </td>
        </tr>
        </table>

</body>
</html>
<?php
```

```
$setViewProjectsDAO = new SetViewProjectsDAO;
$viewShow = $setViewProjectsDAO->find_byId($_POST['viewProjectId']);

?>
<table class="show" rules='rows'>
      <tr>
            <th bgcolor="#000000" colspan=4>
                  <form name='RMT' method='post' action='<?print
$PHP_SELF?>'>
                  <font color=white><b>..::Set Projects to
View::..</b></font>
            </th>
      </tr>
        <tr>
                  <td>
                  <b>Database Location</b>
            </td>
                  <td>
                  <input type="text" name="location" value="<?print
$viewShow->get_location()?>">
            </td>
      </tr>
        <tr>
                  <td>
                  <b>Company Name</b>
            </td>
                  <td>
                  <input type="text" name="companyName" value="<?print
$viewShow->get_companyName()?>">
            </td>
      </tr>
        <tr>
                  <td>
                  <b>Project Name</b>
            </td>
                  <td>
                  <input type="text" name="projectName"  value="<?print
$viewShow->get_projectName()?>">
            </td>
      </tr>
        <tr>
                  <td>
                  <b>Database Type</b>
            </td>
                  <td>
                  <select name="dbType">
                 <option value="1" <?if ($viewShow->get_dbType() == "1")
print "selected";?> >mySQL</option>
                 <option value="2" <?if ($viewShow->get_dbType() == "2")
print "selected";?> >Access</option>
                 <option value="3" <?if ($viewShow->get_dbType() == "3")
print "selected";?> >Oracle</option>
                  </select>
            </td>
      </tr>
        <tr>
                  <td>
```

```
                    <b>Database Name</b>
              </td>
                    <td>
                    <input type="text" name="databaseName"  value="<?print
$viewShow->get_databaseName()?>">
              </td>
        </tr>
          <tr>
                    <td>
                    <b>Database User Name</b>
              </td>
                    <td>
                    <input type="text" name="userName"  value="<?print
$viewShow->get_userName()?>">
              </td>
        </tr>
          <tr>
                    <td>
                    <b>Database Password</b>
              </td>
                    <td>
                    <input type="password" name="password"  value="<?print
$viewShow->get_password()?>">
              </td>
        </tr>
        <tr>
                <td colspan=2>
                    <input type="hidden" name="op"
value="submitViewProjects">
                    <input type="hidden" name="userId" value="<?print
$userShow->get_userId()?>">
                    <input type="hidden" name="viewProjectId"
value="<?print $viewShow->get_Id()?>">
                    <input type="submit" value="Submit">

              </td>
        </tr>
        </table>

</body>
</html>
```

APPENDIX C

SOURCE CODE FILES

```
<?php
class SetViewOnlyAction extends AdminController{

var $user;
var $userDAO;

function SetViewOnlyAction(){}

function accept($post){

      if($post == null){
            return "acceptUser.inc";
      }

      $this->userDAO = new UserDAO;
      $this->userSelect = $this->userDAO->get_user($post['id']);
      $this->userShow = $this->userSelect;

      $this->user = new User;
      $this->user->set_userid($this->userShow->get_userid());
      $this->user->set_username($this->userShow->get_username());
      $this->user->set_password($this->userShow->get_password());
      $this->user->set_firstName($this->userShow->get_firstName());
      $this->user->set_lastName($this->userShow->get_lastName());
      $this->user->set_middleInitial($this->userShow-
>get_middleInitial());
      $this->user->set_email($this->userShow->get_email());
      $this->user->set_phone($this->userShow->get_phone());
      $this->user->set_projectName($this->userShow->get_projectName());
      $this->user->set_valid('1');

      if($post['setView'] == "on"){
            $this->user->set_type("V");
      }else{
            $this->user->set_type("N");
      }
      $this->userDAO = new UserDAO;
      if($this->userDAO->update($this->user) == 1){
            $this->confirmation = $this->userDAO->confirmation;
            $subject = "ReMoTe Registration Approved -- View Only";
            $message = "User: ".$this->user->get_firstName()." ".$this-
>user->get_lastName().",\n\n"
."Hello,\n\n"
."Thank you for registering for the ReMoTe.\n"
."This is a response to give your new username and password.\n"
."Please keep this in your records. You will only have viewing\n"
."priviledges.\n"
."\t username=".$this->user->get_username()."\n"
."\t password=".$this->user->get_password()."\n"
."\n\n Please do not respond to this e-mail. If you have any questions"
."please contact your ReMoTe administrator or your manager.";
            $email="remotemaster@rmt.ias.csusb.edu";
            $mail_params = "-f$email"; // set sender to email
            $mail_headers =
                "Return-path: $email\r\n" .
                "Bcc: $email\r\n";
      if (mail($this->user->get_email(), $subject, $message, $headers,
```

```
$mail_params) ) {
        $this->confirmation .= "Mail sent to ".$this->user->get_email();
        } else {
            $this->error_output .= "Mail was <strong>not</strong> sent
to ".$this->user->get_email();
        }
            return "acceptUser.inc";
      }
      else {
            $this->error_output = "Database Error";
            return "acceptUser.inc";
      }
}
}//SetViewOnlyAction
?>
```

```
<?php
class SetViewProjectsAction extends InterfaceController
{
var $confirmation;
var $error_output;

function SetViewProjectsAction(){}

function setViewProjects($post){

    $error = 0;
    if($post == null){
            return "setViewProjects.inc";
    }
            $setViewProjects = new SetViewProjects;
            $setViewProjectsDAO = new SetViewProjectsDAO;
    if ($post['type'] == "delete"){
            if($setViewProjectsDAO->delete($post['viewProjectId']) ==
1){
               $this->confirmation = "<center>Project Location is
Updated. <br>";
               return "intro.inc";
            }else { $this->error_output = "Database Error";
               return "setViewProjects.inc";
            }
    }
    if ($post['type'] == "edit"){
            return "editViewProjects.inc";
    }
    $post['projectName'] = strtoupper($post['projectName']);
    $setViewProjects->set_Id($post['viewProjectId']);
    $setViewProjects->set_location($post['location']);
    $setViewProjects->set_companyName($post['companyName']);
    $setViewProjects->set_projectName($post['projectName']);
    $setViewProjects->set_dbType($post['dbType']);
    $setViewProjects->set_databaseName($post['databaseName']);
    $setViewProjects->set_userName($post['userName']);
    $setViewProjects->set_password($post['password']);
    $setViewProjects->set_personId($post['userId']);

    if($setViewProjectsDAO->create($setViewProjects) == 1){
            $this->confirmation = "<center>Project Location is Updated.
<br>";
            unset($setViewProjects);
            unset($setViewProjectsDAO);
            return "intro.inc";
    }else {

            $this->error_output = "Database Error";
            return "setViewProjects.inc";

            }
}//login_user

}//LoginAction

?>
```

```
<?
class ConnectDAO {
/////////private variables//////////////////////
var $company_name;
var $host_name;
var $user_name;
var $password;
var $database_name;
var $db_type;
var $result;
var $cid;
var $output;
var $db;
var $fetched_once;
var $oracle_sid;
var $parse;

function ConnectDAO(){
include "./settings/setup.conf";

      $this->company_name = $companyname;
      $this->host_name = $host;
      $this->user_name = $usr;
      $this->password = $pwd;
      $this->database_name = $db;
      $this->db_type = $db_type;

      if($_SESSION['dbLocation'] != null && $_SESSION['dbId'] != null){
            $this->company_name = $_SESSION['dbLocation']-
>get_companyName();
            $this->host_name = $_SESSION['dbLocation']->get_location();
            $this->user_name = $_SESSION['dbLocation']->get_userName();
            $this->password = $_SESSION['dbLocation']->get_password();
            $this->database_name = $_SESSION['dbLocation']-
>get_databaseName();
            $this->db_type = $_SESSION['dbLocation']->get_dbType();
      }
      if($this->db_type == '2'){
            $this->db = new ODBCSocketServer;
      }
}//ConnectDAO

function connect(){
      if ($this->db_type == '1'){
            $this->cid = mysql_pconnect($this->host_name, $this-
>user_name, $this->password);
      }else if ($this->db_type == '2'){
            $this->db->connect($this->host_name, $this->user_name,
$this->password);
            $this->db->select_db($this->database_name);
      }else if ($this->db_type == '3'){
            $db ="(DESCRIPTION =
                        (ADDRESS =
                            (PROTOCOL = TCP)
                            (HOST = $this->host_name)
                            (PORT = 1521)
                         )
```

```
                        (CONNECT_DATA = (SID = oracle))
                    )";
            $this->cid = ociplogon($this->user_name, $this->password,
$db);
}//connect

function connectTest($dbArray){
      if ($dbArray->get_dbType() == '1'){
            $test = @mysql_pconnect($dbArray->get_location(), $dbArray-
>get_userName(), $dbArray->get_password());
            @mysql_close($test);
      }else if ($dbArray->get_dbType() == '2'){
            $socket = @fsockopen($dbArray->get_location(), 9628, $errno,
$error, 5);
            if(!$socket)
                  return false;
            return true;
      }else if($dbArray->get_dbType() == '3'){
            $db ="(DESCRIPTION =
                        (ADDRESS =
                            (PROTOCOL = TCP)
                            (HOST = ".$dbArray->get_location().")
                            (PORT = 1521)
                         )
                        (CONNECT_DATA = (SID = ".$dbArray-
>get_databaseName().")
                    )";
                        $test = @ociplogon($dbArray->get_userName(),
$dbArray->get_password(), $db);
                    OCILogOff($test);
  }
  return $test;

}//connectTest

function set_query($SQL) {
      if ($this->db_type == '1'){
            $SQL = ereg_replace("table", "table if not exists", $SQL);
           $result = @mysql_db_query($this->database_name, $SQL, $this-
>cid);
      }
      else if ($this->db_type == '2'){
          $SQL = ereg_replace("INT NOT NULL AUTO_INCREMENT PRIMARY KEY",
"autoincrement(0, 1)", $SQL);
          if ((preg_match('/ user /', $SQL)==1)){
              $SQL = ereg_replace("username", "a_username", $SQL);
              $SQL = ereg_replace("password", "b_password", $SQL);
              $SQL = ereg_replace("firstName", "c_firstName", $SQL);
              $SQL = ereg_replace("lastName", "d_lastName", $SQL);
               $SQL = ereg_replace("middleInitial", "e_middleInitial",
$SQL);
              $SQL = ereg_replace("phone", "f_phone", $SQL);
              $SQL = ereg_replace("email", "g_email", $SQL);
              $SQL = ereg_replace("role", "h_role", $SQL);
              $SQL = ereg_replace("projectName", "i_projectName", $SQL);
              $SQL = ereg_replace("valid", "j_valid", $SQL);
              $SQL = ereg_replace("address", "k_address", $SQL);
```

```
            $SQL = ereg_replace("picture", "l_picture", $SQL);
          }if((preg_match('/ setNumberPhases /', $SQL)==1)){
            $SQL = ereg_replace("numberPhases", "a_numberPhases",
$SQL);//setNumberPhasesDAO
            $SQL = ereg_replace("userId", "b_userId", $SQL);
            $SQL = ereg_replace("iteration", "c_iteration", $SQL);
          }if((preg_match('/ selectTeam /', $SQL)==1)){
              $SQL = ereg_replace("managerId", "a_managerId",
$SQL);//SelectTeamDAO
            $SQL = ereg_replace("subUserId", "b_subUserId", $SQL);
            $SQL = ereg_replace("projectName", "c_projectName", $SQL);
            $SQL = ereg_replace("iteration", "d_iteration", $SQL);
          }if((preg_match('/ setTeamNames /', $SQL)==1)){
            $SQL = ereg_replace("companyName", "a_companyName",
$SQL);//setTeamNamesDAO
            $SQL = ereg_replace("projectName", "b_projectName", $SQL);
            $SQL = ereg_replace("iteration", "c_iteration", $SQL);
            $SQL = ereg_replace("teamId", "d_teamId", $SQL);
          }if((preg_match('/ threads /', $SQL)==1)){
            $SQL = ereg_replace("companyName", "a_companyName",
$SQL);//AddThreadDAO
            $SQL = ereg_replace("iteration", "b_iteration", $SQL);
            $SQL = ereg_replace("projectName", "c_projectName", $SQL);
            $SQL = ereg_replace("fileName", "d_fileName", $SQL);
            $SQL = ereg_replace("fileType", "e_fileType", $SQL);
            $SQL = ereg_replace("fileSize", "f_fileSize", $SQL);
            $SQL = ereg_replace("versionNumber", "g_versionNumber",
$SQL);
            $SQL = ereg_replace("userId", "h_userId", $SQL);
            $SQL = ereg_replace("phaseId", "i_phaseId", $SQL);
          }if((preg_match('/ fileNotes /', $SQL)==1)){
            $SQL = ereg_replace("userId", "a_userId", $SQL);
            $SQL = ereg_replace("projectName", "b_projectName", $SQL);
            $SQL = ereg_replace("date", "c_date", $SQL);
            $SQL = ereg_replace("fileId", "d_fileId", $SQL);
            $SQL = ereg_replace("iteration", "e_iteration", $SQL);
            $SQL = ereg_replace("message", "f_message", $SQL);
          }if((preg_match('/ messageBoard /', $SQL)==1)){
            $SQL = ereg_replace("userId", "a_userId", $SQL);
            $SQL = ereg_replace("projectName", "b_projectName", $SQL);
            $SQL = ereg_replace("date", "c_date", $SQL);
            $SQL = ereg_replace("broadcast", "d_broadcast", $SQL);
            $SQL = ereg_replace("replyId", "e_replyId", $SQL);
            $SQL = ereg_replace("subject", "f_subject", $SQL);
          }if((preg_match('/ setDates /', $SQL)==1)){
            $SQL = ereg_replace("phaseNumber", "a_phaseNumber",
$SQL);//setDatesDAO
            $SQL = ereg_replace("companyName", "b_companyName", $SQL);
             $SQL = ereg_replace("projectName", "c_projectName", $SQL);
             $SQL = ereg_replace("phaseName", "d_phaseName", $SQL);
             $SQL = ereg_replace("startYear", "e_startYear", $SQL);
             $SQL = ereg_replace("startMonth", "f_startMonth", $SQL);
             $SQL = ereg_replace("startDay", "g_startDay", $SQL);
             $SQL = ereg_replace("phaseDays", "h_phaseDays", $SQL);
             $SQL = ereg_replace("iteration", "i_iteration", $SQL);
             $SQL = ereg_replace("userId", "j_userId", $SQL);
          }if((preg_match('/ settings /', $SQL)==1)){
```

```
                $SQL = ereg_replace("companyName", "a_companyName", $SQL);
                $SQL = ereg_replace("projectName", "b_projectName", $SQL);
                $SQL = ereg_replace("bugzilla", "c_bugzilla", $SQL);
                $SQL = ereg_replace("message", "d_message", $SQL);
                $SQL = ereg_replace("numberIterations",
"e_numberIterations", $SQL);
            }if((preg_match('/ setViewProjects /', $SQL)==1)){
                $SQL = ereg_replace("location", "a_location", $SQL);
                $SQL = ereg_replace("companyName", "b_companyName", $SQL);
                $SQL = ereg_replace("projectName", "c_projectName", $SQL);
                $SQL = ereg_replace("dbType", "d_dbType", $SQL);
                $SQL = ereg_replace("userName", "e_userName", $SQL);
                $SQL = ereg_replace("password", "f_password", $SQL);
                $SQL = ereg_replace("personId", "g_personId", $SQL);
                $SQL = ereg_replace("databaseName", "h_databaseName",
$SQL);
            }if((preg_match('/ statistics /', $SQL)==1)){
                $SQL = ereg_replace("iteration", "a_iteration", $SQL);
                $SQL = ereg_replace("companyName", "b_companyName", $SQL);
                $SQL = ereg_replace("projectName", "c_projectName", $SQL);
                $SQL = ereg_replace("percentage", "d_percentage", $SQL);
                $SQL = ereg_replace("userId", "e_userId", $SQL);
                $SQL = ereg_replace("date", "f_date", $SQL);
                $SQL = ereg_replace("totalCorrect", "g_totalCorrect",
$SQL);
                $SQL = ereg_replace("numberPhases", "h_numberPhases",
$SQL);
            }
            $result = $this->db->query($SQL);
        }
    else if ($this->db_type == '3'){
            if (preg_match('/create/', $SQL)){
                $oracleCreate ="create sequence test_seq start with 1
increment by 1 nomaxvalue";
                $this->parse = ociparse($this->cid, $oracleCreate);
                $result = @ociexecute($this->parse,
OCI_COMMIT_ON_SUCCESS);
            }

            $SQL = ereg_replace("double", "NUMBER", $SQL);
            if (!preg_match('/into/', $SQL)){
                $SQL = ereg_replace(" int", " NUMBER", $SQL);
            }

            $SQL = ereg_replace("INT NOT NULL AUTO_INCREMENT PRIMARY
KEY", " NUMBER", $SQL);
            $SQL = ereg_replace("text", "VARCHAR2(240)", $SQL);
            $SQL = ereg_replace("user", "oracleUser",  $SQL);
            if (!preg_match('/pdate/', $SQL)){
                $SQL = ereg_replace("date", "oracleDate",  $SQL);
            }
            if ((preg_match('/insert into/', $SQL)) AND
(!preg_match("/url/", $SQL))){
                $output = explode("(", $SQL);
                $step2 = "(test_seq.nextval,".$output[2];
                $temp = $output[0] . "values" . $step2;
                $this->parse = @ociparse($this->cid, $temp);
```

```
                $result = @ociexecute($this->parse,
OCI_COMMIT_ON_SUCCESS);
                return $result;
            }
            $this->parse = @ociparse($this->cid, $SQL);
                $result = @ociexecute($this->parse,
OCI_COMMIT_ON_SUCCESS);
  }
  unset ($SQL);
      return $result;
}//set_query

function fetch_row($result){
      if($this->db_type == '1'){
            $output = @mysql_fetch_row($result);
      }
      else if ($this->db_type == '2'){
            $output = $this->db->fetch_row();
      }
  else if ($this->db_type == '3'){
            $out = array();
            $newarr = array();
            $output = @ocifetchinto($this->parse, $out, OCI_ASSOC);
                                    $output = $out;
            $tmparr = $output;
            $i=0;
            foreach ($tmparr as $v) {
                $newarr[$i] = $v;
                $i++;
            }
            unset ($out);
            unset ($output);
            unset ($tmparr);
            return $newarr;
      }
      return $output;
}//fetch_row

function next_row($result){
      if($this->db_type == '1'){
            $output = @mysql_fetch_row($result);
      }
      else if ($this->db_type == '2'){
            $output = $this->db->next_row();
            $output = $this->db->fetch_row();
      }
      else if ($this->db_type == '3'){
            $out = array();
            $newarr = array();
            $output = @ocifetchinto($this->parse, $out, OCI_ASSOC);
            $output = $out;
            $tmparr = $output;
            $i=0;
            foreach ($tmparr as $v) {
               $newarr[$i] = $v;
               $i++;
            }
```

```
            unset ($out);
            unset ($output);
            unset ($tmparr);
            return $newarr;
      }
      return $output;
}//next_row

function num_rows($result){
  if($this->db_type == '1'){
            $output = @mysql_num_rows($result);
  }
  else if ($this->db_type == '2'){
            $output = $this->db->num_rows($result);
  }
  else if ($this->db_type == '3'){
        $temp = $this->fetch_row($this->parse);
        $i=0;
        while($temp){
          $i++;
          $temp = $this->fetch_row($this->parse);
        }
        $output = $i;
  }
  return $output;
}//num_row

function error(){
  if($this->db_type == '1'){
            $output = @mysql_error();
  }
  else if ($this->db_type == '2'){
            $output = @$this->db->error();
  }
  else if ($this->db_type == '3'){
            $output = oci_error($this->parse);
  }
  return $output;
}//error

function close(){
      if($this->db_type == '1'){
            @mysql_close($this->cid);
      }
      else if ($this->db_type == '2'){
            $this->db->close();
      }
      else if ($this->db_type == '3'){
            OCILogOff($this->cid);
      }
      unset($this);
}//close

}//class

?>
```

```
<?php
class SetViewProjects
{
//////////private//////////
var $location;
var $personId;
var $projectName;
var $databaseName;
var $company_name;
var $db_type;
var $user_name;
var $password;
var $id;

function SetViewProjects(){}//setDates
function set_Id($id) {$this->id = $id;}
function set_location($location) {$this->location = $location;}
function set_companyName($company_name){$this->company_name =
$company_name;}
function set_databaseName($databaseName){$this->databaseName =
$databaseName;}
function set_projectName($projectName){$this->projectName =
$projectName;}
function set_dbType($dbType) {$this->dbType = $dbType;}
function set_password($password){$this->password = $password;}
function set_userName($user_name){$this->user_name = $user_name;}
function set_personId($personId){$this->personId = $personId;}
function get_Id() {return $this->id;}
function get_location() { return $this->location;}
function get_companyName(){return $this->company_name;}
function get_databaseName(){return $this->databaseName;}
function get_projectName(){return $this->projectName;}
function get_dbType(){return $this->dbType;}
function get_password(){return $this->password;}
function get_userName(){return $this->user_name;}
function get_personId(){return $this->personId;}

function read($post){
      $settings = new SetViewProjects;
      $settings->set_Id($post[0]);
      $settings->set_location($post[1]);
      $settings->set_companyName($post[2]);
      $settings->set_projectName($post[3]);
      $settings->set_dbType($post[4]);
      $settings->set_userName($post[5]);
      $settings->set_password($post[6]);
      $settings->set_personId($post[7]);
      $settings->set_databaseName($post[8]);

      return $settings;
}
}//teamNameclass
?>
```

```
<?
class SetViewProjectsDAO extends SetViewProjects
{
var $company_name;
var $host_name;
var $user_name;
var $password;
var $database_name;
var $confirmation;

function SetViewProjectsDAO(){
/*******************
Get the information from the setup files
*******************/
include "./settings/setup.conf";
$this->company_name = $companyname;
$this->host_name = $host;
$this->user_name = $usr;
$this->password = $pwd;
$this->database_name = $db;
$this->user = new User;
}//database

function create($post){
/*
      Check if Dates already Exist
*/
$db = new ConnectDAO;
$db->connect();
$id = $post->get_Id();
$SQL = "select * from setViewProjects where id=$id";
$result = $db->set_query($SQL);
$exist = $db->fetch_row($result);
if($exist){
      return $this->update($post);
}

$SQL = "create table setViewProjects (";
$SQL .= " ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,";
$SQL .= "location text,";
$SQL .= "companyName text,";
$SQL .= "projectName text,";
$SQL .= "dbType int,";
$SQL .= "userName text,";
$SQL .= "password text,";
$SQL .= "personId int,";
$SQL .= "databaseName text)";
$result = $db->set_query($SQL);
if(!$result) print $db->error();
$this->confirmation = "<li>Table Creation = $result";

$personId = $post->get_personId();
$dbType = $post->get_dbType();

$SQL = "insert into setViewProjects (";
$SQL .= "location,";
$SQL .= "companyName,";
```

```
$SQL .= "projectName,";
$SQL .= "dbType,";
$SQL .= "userName,";
$SQL .= "password,";
$SQL .= "personId,";
$SQL .= "databaseName) values (";
$SQL .= "'".$post->get_location()."',";
$SQL .= "'".$post->get_companyName()."',";
$SQL .= "'".$post->get_projectName()."',";
$SQL .= "$dbType,";
$SQL .= "'".$post->get_userName()."',";
$SQL .= "'".$post->get_password()."',";
$SQL .= "$personId,";
$SQL .= "'".$post->get_databaseName()."')";
$result = $db->set_query($SQL);
$this->confirmation .= "<li>Table Insert = $result";
if(!$result) print $db->error();
$db->close();
unset ($db);
return $result;
}//create

function update($post){
$personId = $post->get_personId();
$dbType = $post->get_dbType();
$SQL = "update setViewProjects set";
$SQL .= " location=";
$SQL .= "'".$post->get_location()."',";
$SQL .= " companyName=";
$SQL .= "'".$post->get_companyName()."',";
$SQL .= " projectName=";
$SQL .= "'".$post->get_projectName()."',";
$SQL .= " dbType=";
$SQL .= "$dbType,";
$SQL .= " userName=";
$SQL .= "'".$post->get_userName()."',";
$SQL .= " password=";
$SQL .= "'".$post->get_password()."',";
$SQL .= " personId=";
$SQL .= "$personId,";
$SQL .= " databaseName=";
$SQL .= "'".$post->get_databaseName()."'";
$SQL .= " where id=";
$SQL .= $post->get_Id();

$db = new ConnectDAO;
$db->connect();
$result = $db->set_query($SQL);
if(!$result) print $db->error();
$this->confirmation .= "<li>Table Update = $result";
if(!$result) print $db->error();
$db->close();
unset ($db);
return $result;

}//update
```

```
function delete($Id){
/*
 deletes the managers settings
*/
$db = new ConnectDAO;
$db->connect();
$result = $db->set_query("delete from setViewProjects where id=$Id");
$db->close();
unset ($db);
return $result;

}//delete

function find_byId($Id){
$db = new ConnectDAO;
$db->connect();
$result = $db->set_query("select * from setViewProjects where id = $Id");
$loop= $db->fetch_row($result);
while($loop){
      $setViewProjects = new SetViewProjects;
      $setup = $setViewProjects->read($loop);
      $loop = $db->next_row($result);
}
$db->close();
unset ($loop);
unset ($db);
return $setup;
}//find_byId

function findAll(){
$db = new ConnectDAO;
$db->connect();
$result = $db->set_query("select * from setViewProjects order by
projectName ASC");
$loop= $db->fetch_row($result);
while($loop){
      $setViewPorjects = new SetViewProjects;
      $setup = $setViewProjects->read($loop);
      $list[] = $setup;
      $loop = $db->next_row($result);
}
mysql_close();
unset ($loop);
unset ($db);
return $list;

}//findAll

function findAllByProject($projectName, $companyName){

$db = new ConnectDAO;
$db->connect();
$result = $db->set_query("select * from setViewProjects where
projectName='$projectName' AND companyName='$companyName' order by
phaseNumber ASC");
$loop= $db->fetch_row($result);
while($loop){
```

```
        $setViewProjects = new SetViewProjects;
        $setup = $setViewProjects->read($loop);
        $list[] = $setup;
        $loop = $db->next_row($result);
}
$db->close();
unset ($loop);
unset ($db);
return $list;

}//findAllByProject

function get_viewProjects($post){
/*
Returns the list the user submitted based on personId and iteration
*/

$db = new ConnectDAO;
$db->connect();
$SQL = "select * from setViewProjects ";
$SQL .= "where personId=".$post->get_userId();
$result = $db->set_query($SQL);

$loop= $db->fetch_row($result);
if(!$loop){
        return 0;
}
else{
        while($loop){
                $setViewProjects = new SetViewProjects;
                $setup[] = $setViewProjects->read($loop);
                $loop = $db->next_row($result);
        }
}
$db->close();
unset ($loop);
unset ($db);
return $setup;

}//get_viewProjects

}//setViewProjectsDAO

?>
```

```
<?php
class URL
{

//////////private//////////
var $url;

function url(){}

function set_url($url) {$this->url = $url;}

function get_url() {return $this->url;}

function read($post){
      $settings = new URL;
      $settings->set_url($post[0]);
      return $settings;
}
}//urlclass

?>


<?
class urlDAO extends URL
{

var $localURL;

function urlDAO(){

$myURL = $_SERVER['SCRIPT_URI'];
$otherURL = $_SERVER['HTTP_HOST'];

if($myURL != null){
      $URL = parse_url($myURL);
      $path = "";
      list($notneeded, $path, $more) = split('[/]', $URL['path']);
      if($URL['port'] != null) {
            $port = ":".$URL['port'];
      }
      $this->localURL =
$URL['scheme']."://".$URL['host'].$port."/".$path."/";

      if($more != null AND !preg_match('[.php]',$more)){
            $this->localURL = $this->localURL.$more."/";
      }
}else{
            list($notneeded, $path, $more) = split('[/]',
$_SERVER['PHP_SELF']);
            $this->localURL = "http://".$otherURL."/".$path."/";

            if($more != null AND !preg_match('[.php]',$more)){
                  $this->localURL = $this->localURL.$more."/";
            }
}
```

```
}//database


function create(){

$db = new ConnectDAO;
$db->connect();
$result = $db->set_query("select * from url");
$exist = $db->fetch_row($result);
if($exist){
      return $this->update();
}

$SQL = "create table url (";
$SQL .= "url text)";
$result = $db->set_query($SQL);

$SQL = "insert into url(";
$SQL .= "url) values (";
$SQL .= "'".$this->localURL."')";
$result = $db->set_query($SQL);
unset ($db);
return $result;
}//create

function update(){

$SQL = "update url set";
$SQL .= " url=";
$SQL .= "'".$this->localURL."'";
$db = new ConnectDAO;
$db->connect();
$result = $db->set_query($SQL);
$this->confirmation .= "<li>Table Update = $result";
unset ($db);
return $result;

}//update

function get_url(){

  $db = new ConnectDAO;
  $db->connect();
      $SQL = "select * from url ";
      $result = $db->set_query($SQL);
      $loop= $db->fetch_row($result);
      if(!$loop){
            return 0;
      }
      else{
            $URL = new URL;
            $setup = $URL->read($loop);
      }
      $db->close();

      unset($URL);
      unset($loop);
```

```
      //print "URL is ".$setup->get_url()."<br>";
      $setup->set_url(ereg_replace("main\.php\/", '', $setup-
>get_url()));
      return $setup->get_url();

}//get_url

}//urlDAO

?>
```

```
<?php
class ODBCSocketServer
{     // Public variables
      var $sHostName;          // Name of the host to connect to
      var $nPort=9628;
      var $sConnectionString; .
      var $sSQL;               // sql query string
      var $sXML="";
      var $sError="";          // Last generated error message
      var $aRecordset=null;
      var $bBOF = true;
      var $bEOF = true;
      var $nRecordCount=0;     // The number of records returned
      var $aFieldNames;        // The names of the fields
      var $nFieldCount=0;      // The number of fields returned
      var $bCaseFolding=true;
      var $_dbName;
      var $_tableName;
      var $_userName;
      var $_password;
      var $_timeout=30;
      var $_errno;
      var $_currentRow = -1;
      var $_result=true;       // return value
      var $_parser;            // xml parser
      var $_currentTag;        // current xml tag
      var $_tmpRow;            // Array of current row values

      function connect($server, $username=null, $password=null)
      {
            list($this->sHostName, $port) = split(':', $server, 2);
            if (isset($port))
                  $this->nPort=$port;
            if (isset($username))
                  $this->_userName=$username;
            if (isset($password))
                  $this->_password=$password;
            return true;
      }

      function _openport() // private function to establish a connection
to the host and return the socket
      {
            $socket = fsockopen($this->sHostName, $this->nPort, &$this-
>_errno, &$error, $this->_timeout);
            if(!$socket)
            {
                  $this->sError = "<?xml version=\"1.0\"?>\r\n"
                        . "<result state=\"failure\">\r\n"
                        . "<error>Connecting to host=$this->sHostName,
port=$this->nPort failed. errno=$errno, error message=$this-
>error</error>\r\n"
                        . "</result>\r\n";
                  $this->_result = false;
            }
            else
            {
```

```
                $this->_result = true;
            }
            return $socket;
        }

        function close()
        {
            unset($this);
            return true;
        }

        function select_db($dbname)
        {
            $this->_dbName = $dbname;
            return true;
        }

        function query($sql){
            $socket = $this->_openport();
            if($socket)
            {
                if (isset($this->_dbName))
                    $this->sConnectionString = "DSN=$this->_dbName;";
                else
                    $this->sConnectionString = "DSN=*;";
                if (isset($this->_userName))
                    $this->sConnectionString .= "UID=$this-
>_userName;";
                if (isset($this->_password))
                    $this->sConnectionString .= "PWD=$this-
>_password;";
                $this->sSQL = HTMLSpecialChars($sql);
                $sSend = "<?xml version=\"1.0\"?>\r\n"
                    . "<request>\r\n"
                    . "<connectionstring>$this-
>sConnectionString</connectionstring>\r\n"
                    . "<sql>$this->sSQL</sql>\r\n"
                    . "</request>\r\n";
                fputs($socket, $sSend);
                $this->sXML = "";
                while (!feof($socket))
                {
                    $this->sXML .= fgets($socket, 128);
                }
                fclose($socket);
                $this->_parseXML();
                $this->_initRecordset();
            }
            else
            {
                $this->_result = false;
            }
            return $this->_result;
        } // query

        function _parseXML()
        {
```

```
            $this->_parser = xml_parser_create();
            xml_set_element_handler($this->_parser, "_startElement",
"_endElement");
            xml_set_character_data_handler($this->_parser,
"_characterData");
            xml_parser_set_option ($this-
>_parser,XML_OPTION_CASE_FOLDING,$this->bCaseFolding);
        xml_set_object($this->_parser,&$this);

        if (!xml_parse($this->_parser, $this->sXML))
            {
                  $this->_result=false;
                  //$this->sError=sprintf("XML ErrorString: %s at line
%d",
                  //
      xml_ErrorString_string(xml_get_ErrorString_code($this->_parser)),
            }
            xml_parser_free($this->_parser);
      }

      function _startElement($parser, $name, $attribs)
      {
            $this->_currentTag = strtolower($name);
            switch ($this->_currentTag)
            {
                  case "row": $this->nFieldCount=0; unset($this-
>_tmpRow); break; // beginning of a row
                  case "column":
                        if ($this->bCaseFolding)
                              $fieldval = $attribs["NAME"];
                        else
                              $fieldval = $attribs["name"];
                        if (isset($fieldval))
                              $this->aFieldNames[$this->nFieldCount] =
$fieldval;
                        break;
                  case "result": $this->nRecordCount=0;
                        if ($this->bCaseFolding)
                              $fieldval = $attribs["STATE"];
                        else
                              $fieldval = $attribs["state"];
                        if ($fieldval == "success")
                              $this->_result = true;
                        else
                              $this->_result = false;
                        break;
                  case "error": break;
            } // switch

      }

      function _endElement($parser, $name)
      {
            $tag = strtolower($name);

            switch($tag)
            {
```

```
                case "row":
                unset( $this->_tmpRow[$this->nFieldCount] );
                $this->aRecordset[$this->nRecordCount++] = $this-
>_tmpRow;
                break; // end of a row, go to next recordset
                case "column": $this->nFieldCount++; break;
                case "result": break; // end of the result
                case "error": break;
          }
          return;
    }

    function _characterData($parser, $data)
    {
    //    if (strlen(trim($data)) == 0) // each data entry is followed
by a "\n", which has strlen(trim($data)) == 0) property
          switch($this->_currentTag)
          {
                case "column":
                      if($data!="\n")
                            $this->_tmpRow[$this->nFieldCount] = $data;
                      else
                            $this->_tmpRow[$this->nFieldCount] = "";
                      break;
                case "row": break;
                case "error":
                      $this->sError = $data;
                      break;
                case "result": break;
          }
    }
    // end XML related functions

    function _initRecordset()
    {
          $this->_currentRow = 0;
          $this->_setCurrentRecord();

    } // _initRecordset()

    function _setCurrentRecord()
    {
          if (0 == $this->nRecordCount)
          {
                $this->bEOF = true;
                $this->bBOF = true;
          }
          else if ($this->_currentRow > ($this->nRecordCount - 1))
          {
                $this->bEOF=true;
                $this->bBOF = false;
                $this->_currentRow = -1;
          }
          else if ($this->_currentRow < 0)
          {
                $this->bBOF=true;
                $this->bEOF=false;
```

```
                $this->_currentRow = -1;
            }
            else
            {
                $this->bEOF = false;
                $this->bBOF = false;
                $record=$this->aRecordset[$this->_currentRow];
                for($i=0;$i<$this->nFieldCount;$i++)
                {
                    $key = $this->aFieldNames[$i];
                    $value = $record[$i];
                    //echo "<pre>key=$key, value=$value\r\n</pre>";
                $this->$key = $value;
            }// for
            }// if

        } // _setCurrentRecord

    // General database functions

        function error() { unset($this); return $this->sError; }

        function errno() { return $this->_errno; }

        function fetch_row()
        {
            if ($this->bBOF || $this->bEOF)
                return false;
            else
                return $this->aRecordset[$this->_currentRow];
        }

        function fetch_assoc()
        {
            if ($this->bBOF || $this->bEOF)
                return false;
            else
            {
                $record=$this->aRecordset[$this->_currentRow];
                for($i=0;$i<$this->nFieldCount;$i++)
                {
                    $key = $this->aFieldNames[$i];
                    $value = $record[$i];
                $assoc[$key] = $value;
            }// for
                return $assoc;
            }
        }

        function num_fields() { return $this->nFieldCount; }

        function field_name_array() { return $this->aFieldNames; }

        function field_name($offset=0) { return $this-
    >aFieldNames[$offset]; }

        function num_rows() { return $this->nRecordCount; }
```

```
	function next_row()
	{
		$this->_currentRow++;
		$this->_setCurrentRecord();
	}

	function prev_row()
	{
		$this->_currentRow--;
		$this->_setCurrentRecord();
	}

	function first_row()
	{
		$this->_currentRow=0;
		$this->_setCurrentRecord();
	}

	function last_row()
	{
		$this->_currentRow = $this->nRecordCount-1;
		$this->_setCurrentRecord();
	}

	function seek_row($row_num=0)
	{
		$this->_currentRow=$row_num;
		$this->_setCurrentRecord();
	}
} // class ODBCSocketServer
?>
```

REFERENCES

[1] IEEE Std. 830-1998 IEEE Recommended Practice of Software Requirements Specifications.

[2] https://rmt.ias.csusb.edu/remote

[3] "Recursive Multi Thread Software Life Cycle Model" Simon Scott, M.S. Thesis, Department of Computer Science, California State University San Bernardino. Dec. 1997.

[4] "The RMT Tool: A Computer Aided Software Engineering Tool for Monitoring and Predicting Software Development Progress" Chung-Ping Lin, M.S. Project, Dept. of Computer Science, CSUSB, December 1998.

[5] "Multi-Database Support in the Recursive Multi-Threaded Software Process Management Tool" Yi-Chiun Kuo, M.S. Project, Dept. of Computer Science, CSUSB, December 2002.

[7] "Managing the Software Development by Using the Recursive Multi- Threaded (RMT) Tool" Arturo I Concepcion, Sunny Lin, Scott J. Simon, IEEE Computer Society, 1999.

[8] "The RMT (Recursive Multi-Threaded) Tool: A Computer Aided Software Engineering Tool for Monitoring and Predicting Software Development Progress" Arturo I Concepcion, Sunny Lin, Scott J. Simon, ACM, 1999.

[9] "Oracle: The Complete reference" George Koch, Kevin Loney, Osborne, 1995.